Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 104
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 104
Espoo 2006 HUT-TCS-A104

AUTOMATA AND LINEAR TEMPORAL LOGIC: TRANSLATIONS WITH TRANSITION-BASED ACCEPTANCE

Heikki Tauriainen

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 104
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 104
Espoo 2006 HUT-TCS-A104

AUTOMATA AND LINEAR TEMPORAL LOGIC: TRANSLATIONS WITH TRANSITION-BASED ACCEPTANCE

Heikki Tauriainen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Computer Science and Engineering, for public examination and debate in Auditorium T2 at Helsinki University of Technology (Espoo, Finland) on the 27th of October, 2006, at 12 o'clock noon.

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietojenkäsittelyteorian laboratorio

Distribution:
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O.Box 5400
FI-02015 TKK, FINLAND
Tel. +358 9 451 1
Fax. +358 9 451 3369
E-mail: lab@tcs.tkk.fi
URL: http://www.tcs.tkk.fi/

© Heikki Tauriainen

ISBN 951-22-8343-3
ISSN 1457-7615

Electronic Edition
September 2006

Abstract: Automata theory provides powerful tools for designing and implementing decision procedures for temporal logics and their applications to the automatic verification of systems against their logical specifications. Implementing these decision procedures by making use of automata built from the systems and their specifications with translation procedures is challenging in practice due to the tendency of the automata to grow easily unmanageably large as the size of the systems or the logical specifications increases.

This thesis develops the theory of translating propositional linear time temporal logic (LTL) into nondeterministic automata via self-loop alternating automata. Unlike nondeterministic automata, self-loop alternating automata are expressively equivalent to LTL and allow a conceptually simple translation of LTL specifications into automata using a set of rules for building automata incrementally from smaller components. The use of generalized transition-based acceptance for automata throughout all constructions gives rise to new optimized translation rules and facilitates designing heuristics for the minimization of automata by making use of language containment tests combined with structural analysis of automata. The generalized definition also supports the translation of self-loop alternating automata into nondeterministic automata by essentially applying the standard subset construction; this construction can be further simplified and optimized when working with automata built from LTL formulas. The translation rules can also be used to identify a syntactic subclass of LTL for which the exponential increase caused by the subset construction in the number of states of the automaton can always be avoided; consequently, the satisfiability problem for this subclass, which is shown to extend related subclasses known from the literature, is **NP**-complete. Additionally, the emptiness of generalized nondeterministic automata is shown to be testable without giving up generalized transition-based acceptance by using a new variant of the well-known nested depth-first search algorithm with improved worst-case resource requirements.

Keywords: linear time temporal logic, alternating automata, nondeterministic automata, transition-based acceptance, minimization, nondeterminization, emptiness checking, nested depth-first search

TIIVISTELMÄ: Automaattiteorian avulla voidaan suunnitella ja toteuttaa temporaalilogiikkojen ratkaisumenetelmiä sekä näiden menetelmien sovelluksia logiikoilla järjestelmistä esitettyjen oikeellisuusvaatimusten tietokoneavusteiseen verifiointiin. Käytännössä näiden ratkaisumenetelmien toteuttaminen kääntämällä järjestelmät ja niiden oikeellisuusvaatimukset automaateiksi on kuitenkin haasteellista, sillä näistä automaateista tulee järjestelmien tai loogisten vaatimusten koon kasvaessa helposti niin suuria, ettei niitä enää voida käsitellä.

Tässä väitöskirjassa kehitetään lineaarisen ajan temporaalilogiikan (LTL) epädeterministisiksi automaateiksi kääntämisen teoriaa käyttämällä käännöksen apuna vain yhden tilan silmukoita sisältäviä alternoivia automaatteja, joilla – toisin kuin epädeterministisillä automaateilla – on sama ilmaisuvoima kuin lineaarisen ajan temporaalilogiikalla. Tätä logiikkaa voidaan kääntää näiksi automaateiksi soveltaen yksinkertaisia sääntöjä automaattien yhdistämiseksi vaiheittain keskenään yhä suuremmiksi automaateiksi. Käyttämällä yleistettyä siirtymäpohjaista hyväksyvyyden määritelmää automaateille kaikissa käännöksen vaiheissa voidaan näin muodostettuja automaatteja sieventää uusin tavoin käyttäen apuna automaattien hyväksymien kielten välisiä sisältyvyyssuhteita sekä automaattien rakenteellisia ominaisuuksia. Yleistetyn määritelmän ansiosta vain yhden tilan silmukoita sisältävät alternoivat automaatit voidaan myös kääntää edelleen epädeterministisiksi automaateiksi soveltamalla yleisesti tunnettua osajoukkokonstruktiota lähes sellaisenaan. Tämä konstruktio voidaan edelleen tehdä yksinkertaisemmin ja tehokkaammin LTL-kaavoista muodostetuille automaateille. Automaattikäännöksessä käytettävien sääntöjen avulla voidaan myös erottaa lineaarisen ajan temporaalilogiikan syntaktinen osajoukko, jonka kaavat on mahdollista kääntää epädeterministisiksi automaateiksi ilman, että automaattien tilojen määrä kasvaa osajoukkokonstruktion tavoin eksponentiaalisesti. Tästä tuloksesta seuraa, että kyseisen LTL:n osajoukon toteutuvuusongelma on **NP**-täydellinen. Osajoukko on samankaltaisia kirjallisuudessa aiemmin esiteltyjä osajoukkoja aidosti laajempi. Väitöskirjassa esitetään myös, kuinka epädeterministisen automaatin hyväksymän kielen tyhjyys voidaan tarkastaa luopumatta yleistetystä siirtymäpohjaisesta hyväksyvyyden määritelmästä käyttäen algoritmia, joka on uusi, huonoimman tapauksen vaatimuksiltaan tehokkaampi muunnos tunnetusta sisäkkäisestä syvyyshakualgoritmista.

AVAINSANAT: lineaarisen ajan temporaalilogiikka, alternoivat automaatit, epädeterministiset automaatit, siirtymäpohjainen hyväksyvyys, automaattien sieventäminen, epädeterminisointi, tyhjyystarkastus, sisäkkäinen syvyyshaku

CONTENTS

PREFACE

This report is the result of my postgraduate studies at the Laboratory for Theoretical Computer Science of Helsinki University of Technology. I wish to thank my advisor, Docent Keijo Heljanko, who originally introduced me into the theory of model checking. Without the many discussions with him my apprehension of many concepts and techniques about this subject would be much poorer. I continue to be amazed by his wealth of ideas and his insight to see the correctness or falsehood of ideas outright without the need to jump into details, which can always be filled in if necessary. Indeed, countless results in this work owe their inclusion directly to his insight. His comments on the numerous drafts of this constantly expanding work (and the time he sacrificed in reviewing it) are much appreciated.

I am thankful also to my supervisor, Prof. Ilkka Niemelä, for his patience and support throughout my postgraduate studies and for letting me be a member of his research group at the Laboratory of Theoretical Computer Science. I thank also Prof. Emeritus Leo Ojala for guiding me in the first steps of my postgraduate studies, in particular, through his challenging and educational seminars. I am grateful also to my colleagues Dr. Sc. (Tech.) Tommi Junttila and Dr. Sc. (Tech.) Timo Latvala for conversations and cooperation, Alexandre Duret-Lutz for discussions on explicit state language emptiness checking algorithms and transition-based acceptance, and to Prof. Orna Kupferman and Dr. Stephan Merz for their thorough pre-examination reviews of this work. Finally, I wish to express my deepest thanks to my parents, without whose tireless support and friendship I would not have had the strength to finish this effort.

This work was supported financially by Helsinki Graduate School in Computer Science and Engineering (HeCSE), Academy of Finland (Project numbers 47754, 53695 and 211025), Finnish Funding Agency for Technology and Innovation (TEKES), Department of Computer Science and Engineering at Helsinki University of Technology, and a personal grant from Nokia Foundation. I wish to thank these institutions for making my full-time postgraduate studies possible.

1 INTRODUCTION

Automata on infinite objects link the theory of reasoning about the correctness of finite-state reactive and concurrent systems to the design of concrete decision procedures for checking the satisfiability or validity of specifications given in formal logic, and for the automatic verification of systems against such specifications [Clarke and Emerson 1982a,b; Queille and Sifakis 1982; Lichtenstein and Pnueli 1985] (a task commonly known as *model checking*). The logical specifications define constraints on the computations of the system, which are seen as infinite trees or sequences of finite sets of truth-valued assertions that record the internal state of the system at discrete consecutive instants of time. The connection between automata and logic arises from the classic interpretation of automata as acceptors of sequences (words) or trees, which are in this case identified with word or tree models of formulas in the logic. For example, testing whether a system meets its specification can be decided by checking that no computation of the system is accepted by a finite automaton that distinguishes exactly those computations that do not satisfy the specification from all possible computations [Vardi and Wolper 1986]. Such an automaton can be obtained automatically from the logical specification by using a *translation procedure* for the logic. This general approach to verification has led to the introduction of a wide variety of automata and corresponding translation and verification procedures designed for capturing the expressive power of many linear and branching time logics (see, for example, [Wolper et al. 1983; Vardi and Wolper 1986; Muller et al. 1988; Emerson and Jutla 1991; Emerson et al. 1993, 2001; Bernholtz et al. 1994; Vardi and Wolper 1994; Vardi 1996; Kupferman et al. 2000, 2001]).

The automatic analysis of structures built from formal descriptions of systems is challenging in practice, because the construction of the structures is extremely sensitive to combinatorial explosion (known as the *state space explosion problem*; see, for example, the survey by Valmari [1998]). This same problem concerns also the translation of specifications into automata, and its severity depends on the expressiveness of the chosen specification logic. The increasing computational complexity of working with increasingly expressive (but still decidable) logics is reflected in automata translation procedures as combinatorial explosion, the worst-case complexity of which can range from polynomial to nonelementary in the length of the logical specification. The struggle against the combinatorial explosion, which limits the size and scope of specifications that can be realistically handled within the resources available in practice, has presented a need for translation procedures that aim to avoid the worst case behavior in as many cases as possible.

This work focuses on automata translation and verification procedures for specifications given as formulas of classic future-time *propositional linear time temporal logic* (LTL) using special classes of *alternating automata on infinite words* with *generalized transition-based acceptance* throughout all constructions. The classic automata-theoretic verification procedure for LTL can be seen as a series of translations as shown in Fig. 1.1. The connection between automata and LTL, this verification procedure, and the concepts used in it are briefly introduced below.

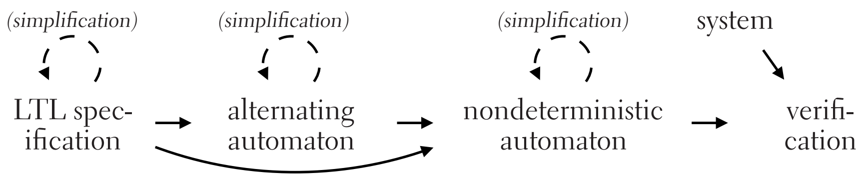


Fig. 1.1: The verification procedure for LTL as a series of translations

Background and Related Work

Logic, automata on infinite objects, and the LTL verification procedure. The connection between logics and automata on infinite objects was first used in the 1960s by Büchi [1962] and Rabin [1969] to prove decidability results for monadic second-order theories of one and many successors, respectively (see, for example, [Thomas 1990], for a survey of the classic results). In the 1980s, these automata were used to obtain decision procedures for modal and temporal logics of programs. Streett [1981, 1982] applied automata to the decision problem of extended propositional dynamic logic; Wolper et al. [1983] and Vardi and Wolper [1994] used automata-theoretic techniques for deciding linear time temporal logics based on the extended temporal logic of Wolper [1981, 1983]; and Emerson and Sistla [1984a,b] proposed an automata-theoretic decision procedure for the branching time temporal logic CTL* introduced by Emerson and Halpern [1983, 1986]. As a special case of their own construction for translating extended temporal logic into automata [Wolper et al. 1983; Vardi and Wolper 1994], Vardi and Wolper [1986] proposed also an explicit decision and verification procedure for *propositional linear time temporal logic* [Pnueli 1977; Gabbay et al. 1980; Pnueli 1981], which supports reasoning about the properties of nonterminating, nonbranching computation paths in a system using invariants, future-time assertions, and qualitative causality and fairness constraints on the occurrence of certain states in the computations. The automata-theoretic verification procedure for this logic (Fig. 1.1) works by translating formulas in the logic in one or more intermediate steps into nondeterministic finite *ω-automata*, i.e., nondeterministic automata on infinite words, whose acceptance is determined, for instance, by a set of designated states that an automaton should visit infinitely often when reacting to its input (a concept known as *Büchi acceptance* after Büchi [1962]). The automata built from formulas are then composed with finite models of systems to compare the behavior of the systems against the logical specifications. The automata-theoretic approach to LTL model checking stimulated active research on techniques for translating LTL and related linear time logics efficiently into nondeterministic automata [Wolper et al. 1983; Michel 1985; Vardi and Wolper 1986; de Jong 1992; Vardi and Wolper 1994; Gerth et al. 1995; Couvreur 1999; Daniele et al. 1999; Somenzi and Bloem 2000; Geilen 2001; Schneider 2001; Wolper 2001; Giannakopoulou and Lerda 2002; Thirioux 2002; Latvala 2003; Sebastiani and Tonetta 2003].

Language emptiness checking and on-the-fly verification. The last step in the verification procedure for LTL in Fig. 1.1 corresponds to checking the set intersection of the computations (i.e., the “language”) of a given system with computations that violate a given LTL specification for *emptiness* by analyzing an automaton obtained by combining the system with a nondeterministic automaton built from the LTL specification. The same technique applies to testing the satisfiability of the logical specification itself, in which case the emptiness analysis is simply done on an automaton built from the specification.

Instead of running the steps in the verification procedure sequentially one after another, the verification procedure can be implemented using *on-the-fly* translation and emptiness checking algorithms [Courcoubetis et al. 1991, 1992; Gerth et al. 1995; Couvreur 1999; Hammer et al. 2005], which allow running several steps of the procedure in an interleaved fashion without the need to finish all previous phases of the procedure before proceeding to the next one. This interleaving of the verification steps is often essential in practice in trying to find a violation of the given logical specification as quickly as possible without generating and storing all intermediate results, which may have prohibitively high resource requirements in the worst case. The state space explosion problem has encouraged active research, in particular, on efficient on-the-fly search algorithms for checking the set of words accepted by a nondeterministic ω-automaton for emptiness. Although this problem is in principle straightforward to solve using standard algorithms for detecting cycles in a directed graph, much effort has nevertheless been put on optimizing the performance of these algorithms and reducing their memory usage in practical applications. The two main lines of state-of-the-art explicit state emptiness checking algorithms for ω-automata using Büchi acceptance can be divided into algorithms that analyze the maximal strongly connected components of automata [Couvreur 1999; Couvreur et al. 2005; Geldenhuys and Valmari 2004, 2005; Hammer et al. 2005]—usually, by extending and optimizing the well-known algorithm of Tarjan [1971, 1972]—and algorithms based on the *nested depth-first search* introduced by Courcoubetis et al. [1991, 1992] (see, for example, [Godefroid and Holzmann 1993; Holzmann et al. 1997; Bošnački 2002, 2003; Gastin et al. 2004; Tauriainen 2004, 2006; Schwoon and Esparza 2005; Couvreur et al. 2005]).

Alternating automata. A useful observation on understanding the connection between logics and automata was made by Muller et al. [1988], who showed temporal logics to be naturally translatable into subclasses of *alternating automata* [Kozen 1976; Chandra and Stockmeyer 1976; Brzozowski and Leiss 1980; Chandra et al. 1981] instead of the combinatorially more complex nondeterministic automata. Alternation combines the nondeterministic (existential) choice for the "next" state of an automaton with universal choice to allow an automaton to enter possibly several "next" states at once while reading its input by spawning copies of itself that work independently on the rest of the input. Muller et al. [1988] demonstrated their approach with a translation from a branching time version of the extended temporal logic of Wolper [1981, 1983] to *weak* alternating automata [Muller et al. 1986, 1992]. As opposed to constructions for translating logics directly into nondeterministic automata, the size of which may be exponential in the length of the logical specifications, the opportunity to mix existential and universal choice in automata gives rise to translation procedures which yield automata that have linear size in the length of the specifications. This connection between logic and alternating automata led to the introduction of another line of translation procedures from LTL and its extensions into automata [Isli 1994, 1996; Vardi 1994; Rohde 1997; Manna and Sipma 2000; Gastin and Oddoux 2001, 2003; Fritz 2003; Tauriainen 2003; Hammer et al. 2005].

The seemingly obvious advantage of using alternating automata in place of nondeterministic automata to avoid combinatorial explosion in verification comes with a cost, however, since the extra succinctness in representation prevents working with alternating automata using the same algorithmic techniques that apply, for example, to language emptiness checking for nondeterministic automata. This difficulty is traditionally overcome by first translating alternating automata into nondeterministic ones by applying one of the *nondeterminization constructions* proposed in the literature for alternating automata on infinite words [Miyano and Hayashi 1984a,b; Lindsay 1988; Isli 1994, 1996; Rohde 1997; Gastin and Oddoux 2001; Fritz 2003; Hammer et al. 2005; Fritz 2005] or trees [Muller et al. 1986, 1992; Emerson and Jutla 1991; Muller and Schupp 1995]. Even though the exponential worst-case combinatorial cost of nondeterminization appears to void any advantages gained by translating LTL first into alternating automata, alternating automata have nevertheless been argued to provide useful additional insight into translation procedures from LTL into automata [Vardi 1995]. For example, even though nondeterministic automata are in general strictly more expressive than LTL, direct translation procedures from LTL into automata do not usually concern themselves with any special properties of the constructed automata, missing a possible correspondence between LTL and a subclass of nondeterministic automata. On the other hand, similarly to the equally expressive formalisms such as the *first-order theory of linear order* (whose expressiveness coincides further with *star-free ω-languages* [Thomas 1979], and *counter-free ω-automata* [Thomas 1981]), LTL has been shown [Rohde 1997; Löding and Thomas 2000] to be expressively equivalent to a simple subclass of alternating automata known as *very weak* [Isli 1994, 1996; Rohde 1997; Gastin and Oddoux 2001], *linear* [Löding and Thomas 2000], or *linear weak* [Merz and Sezgin 2003; Hammer et al. 2005] alternating automata. This strong correspondence between logic and automata allows to optimize constructions for this special class of automata with techniques that do not necessarily apply to the strictly more expressive nondeterministic or alternating ω-automata, whose expressiveness matches that of the *monadic second-order theory of linear order*, or, equivalently, *regular ω-languages*, both in the nondeterministic [Büchi 1962; McNaughton 1966] and the alternating case [Miyano and Hayashi 1984a,b; Lindsay 1988].

Generalized transition-based acceptance. In practice, the implementation of the verification procedure shown in Fig. 1.1 involves many decisions, such as choosing definitions to be used for the underlying automata. To maximize the efficiency of an implementation, the chosen definitions should facilitate expressing the automata succinctly (say, using as few states as possible) while still allowing efficient manipulation of the automata. Even simple changes in the definitions are known to affect the opportunities for minimizing the number of states in the automata: examples include the choice between single or multiple initial states, and various generalizations of the notion of acceptance. For example, instead of specifying acceptance in an automaton using a single set of designated "accepting" states that the automaton should visit infinitely often, many automata translation procedures for LTL define acceptance using a *family* of state sets, all of which should be visited infinitely often to make the automaton accept its input. This notion of *generalized Büchi acceptance* was originally introduced to facilitate the expression of simple liveness requirements—previously studied, e.g., in branching time temporal logic verification [Clarke et al. 1983, 1986]—on the behavior of systems modeled as synchronizing automata [Aggarwal et al. 1990]. Gerth et al. [1995] later used generalized acceptance as a conceptual aid in their translation procedure from LTL into nondeterministic automata for matching syntactic properties of LTL formulas directly with sets of accepting states. Unlike previous constructions, which defined automata as tableaux of the worst-case exponential size, the on-the-fly construction of Gerth et al. [1995], which can itself be seen to be based on earlier work on tableau methods for branching [Ben-Ari et al. 1981, 1983; Clarke and Emerson 1982a,b] and linear time [Wolper 1981, 1983; Manna and Wolper 1982, 1984; Wolper 1985] logics, provided an explicit procedure for constructing only the actually relevant part of an automaton. This feature of the construction made it a popular source of many related translation procedures, from direct improvements to the explicit tableau construction [Daniele et al. 1999; Somenzi and Bloem 2000; Wolper 2001; Giannakopoulou and Lerda 2002; Thirioux 2002; Sebastiani and Tonetta 2003] to translations geared towards a symbolic representation of the automata [Couvreur 1999; Schneider 2001].

The multiple sets of accepting states also have a direct impact on the succinctness of the representation of automata, even though the additional sets do not add to the expressive power of the automata [Emerson and Sistla 1984a,b; Courcoubetis et al. 1991, 1992]. Although still more succinct representations are possible through further generalizations of acceptance with no change in the expressiveness of the automata (see, for example, [Thomas 1997], for a survey of various classic notions of acceptance), such alternative notions have been only rarely used in the context of translating LTL into automata [Michel 1985; de Jong 1992].

Acceptance can also be generalized by associating it with the *transitions* instead of the states of an automaton [Couvreur 1999; Gastin and Oddoux 2001; Giannakopoulou and Lerda 2002; Thirioux 2002; Tauriainen 2003]. As in the case of moving from one to many acceptance sets, this simple change in the notion of acceptance does not add to the expressiveness of the automata due to the interreducibility of state-based and transition-based acceptance (see, for example, Chapter 1 of the textbook [Perrin and Pin 2004]). However, the transition-based notion for acceptance is again more succinct: even though state-based acceptance can be reduced to transition-based acceptance without adding any new states or transitions to an automaton—visiting a state infinitely often implies taking a transition leaving the state infinitely often—the same does not hold for the converse reduction in the general case. It can thus be said that transition-based acceptance generalizes state-based acceptance (see, e.g., [Giannakopoulou and Lerda 2002] for

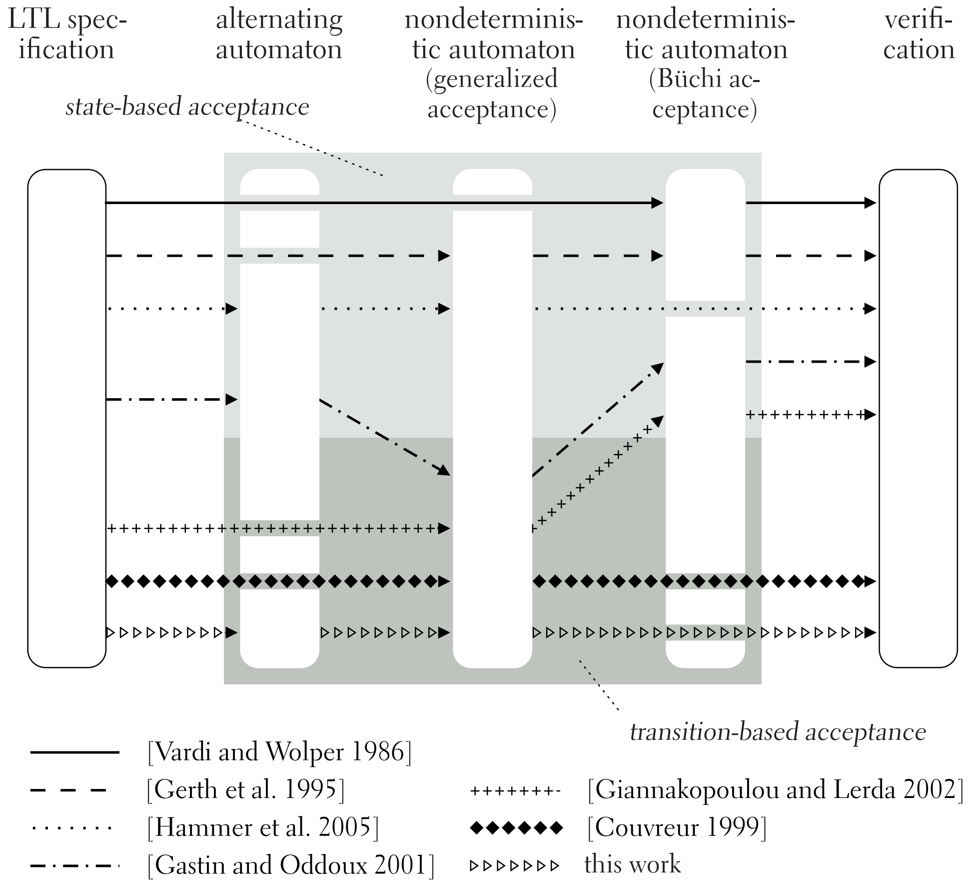


Fig. 1.2: Differences between automata-theoretic LTL verification procedures suggested in the literature

examples). A similar asymmetry concerns the placement of labels, i.e., the symbols that the automaton reads from its input, in the automata; again, replacing single symbols with sets of symbols, or placing the labels on transitions instead of states lead to more succinct definitions for automata.

The use of nondeterministic automata with generalized transition-based acceptance for LTL verification was proposed by Couvreur [1999] and later advocated by many other authors due to its simple benefits for minimizing the representation of automata [Gastin and Oddoux 2001; Thirioux 2002; Giannakopoulou and Lerda 2002; Duret-Lutz and Poitrenaud 2004]. Tauriainen [2003] considered extending the transition-based approach to alternating automata. To take the best possible advantage of these automata in the verification procedure, each verification step should preferably be implemented with algorithms that are able to work directly with this definition to avoid spending additional effort on converting between expressively equivalent formalisms. Surprisingly, only few constructions [Couvreur 1999; Tauriainen 2003; Hammer et al. 2005] strive to achieve this goal fully in practice: most other translations suggested in the literature employ additional conversions between formalisms to finally obtain automata with a classic Büchi acceptance condition specified using a set of states. Figure 1.2 illustrates some of these conceptual differences between several automata-theoretic LTL verification procedures suggested in the literature. For simplicity, we list only a single representative reference to literature on the various approaches.

Minimization of automata. The verification procedure for LTL has also raised interest in techniques for the minimization of ω-automata in general to counter combinatorial explosion. In addition to techniques that exploit special structural properties of automata [Rohde 1997; Etessami and Holzmann 2000; Somenzi and Bloem 2000; Gastin and Oddoux 2001; Thirioux 2002], constructions based on various *simulation relations* have also been proposed for the minimization of both nondeterministic [Etessami and Holzmann 2000; Somenzi and Bloem 2000; Etessami et al. 2001, 2005; Etessami 2002; Gurumurthy et al. 2002] and alternating automata [Fritz and Wilke 2002, 2005; Fritz 2003]. On the other hand, the translation of LTL into automata has been improved further by making use of syntactic techniques such as simplifying LTL specifications by rewriting [Etessami and Holzmann 2000; Somenzi and Bloem 2000; Thirioux 2002]. These optimizations appear in Fig. 1.1 as the dashed loops "inside" each translation phase. Syntactic optimization techniques have been applied also to the translation steps between phases [Daniele et al. 1999; Giannakopoulou and Lerda 2002].

Symbolic tableau procedures. In addition to the explicit automata-based approach to LTL verification, there exist also verification methods that perform their task using implicit "symbolic" representations of systems and *tableaux* built from logical specifications [Clarke et al. 1994, 1997; Kesten et al. 1998]. As a matter of fact, procedures suggested for the construction of such tableaux [Lichtenstein and Pnueli 1985, 2000; Burch et al. 1992; Kesten et al. 1993; Clarke et al. 1994, 1997; Kesten et al. 1998] can easily be seen as another line of automata translation procedures by identifying the nodes in a tableau directly with the states of a nondeterministic automaton. However, due to the implicit representation used for the connections between tableau nodes (which can analogously be identified with transitions of a corresponding automaton), the symbolic tableau constructions are not usually concerned with questions such as the minimization of the number of nodes in the tableaux.

Extensions to other logics. Although most translation procedures from LTL into automata—including the one that will be presented in this work—concentrate only on future time, also constructions for LTL extended with past time connectives have been proposed in the literature, using both nondeterministic [Vardi and Wolper 1986; Ramakrishna et al. 1992b; Schneider 2001] and alternating automata [Manna and Sipma 2000; Kupferman et al. 2001; Gastin and Oddoux 2003] in the translation. Extensions to other specification formalisms include constructions for branching time logics [Bernholtz et al. 1994; Kupferman et al. 2000], logics augmented with extended operators [Vardi 1988; Vardi and Wolper 1994; Kupferman et al. 2001; Laroussinie et al. 2002; Bustan et al. 2005] or first-order quantification [Etessami 1999], the propositional μ-calculus [Emerson and Jutla 1991; Emerson et al. 1993, 2001], interval logics [Ramakrishna et al. 1992a, 1996], and temporal logics on infinite *traces* instead of words [Gastin et al. 1998; Bollig and Leucker 2001, 2003]. On the other hand, special constructions targeted towards efficient automata translation of LTL safety properties have also been proposed [Geilen 2001; Latvala 2003], together with constructions that aim to reduce nondeterministic choice between transitions in automata [Thirioux 2002; Sebastiani and Tonetta 2003].

Organization and Contributions of This Work

This work presents a unified approach to translating future-time LTL into automata and checking for the emptiness of the automata by making use of special cases of a single definition of alternating ω-automata with generalized transition-based acceptance throughout all constructions, advocating (as many authors before) this type of acceptance as a concept well-suited for both understanding and implementing the verification procedure. Some of the presented results have previously appeared in [Tauriainen 2003, 2004, 2005, 2006].

The chosen type of generalized acceptance, which combines an "inverse" interpretation of classic nongeneralized Büchi acceptance (previously used in LTL-to-automata translation procedures by Gastin and Oddoux [2001]) with the intuitive connection between multiple "acceptance conditions" and syntactic properties of LTL formulas [Gerth et al. 1995], is used to define a translation procedure in which the introduction of new acceptance conditions is completely transparent. Another technique used for simplifying the presentation is to adopt a definition of alternating automata which supports direct representation of individual *transitions*, which are essential, for example, for depicting automata as traditional state graphs; the concept of a transition is also convenient for designing and explaining simplification techniques for alternating automata. Similarly, a definition of runs of alternating automata is used that allows many standard theoretical constructions on runs of alternating automata (for example, obtaining a finite representation for the "levels" of a run after uniformization; see, for example, [Muller and Schupp 1995]) to be viewed as direct transformations on runs of alternating automata. The basic definitions of infinite words, linear time temporal logic and automata are reviewed in Ch. 2. This chapter also introduces *self-loop alternating automata*, which can be seen as another "transition-based" version of a subclass of alternating automata known as *very weak alternating* [Isli 1994, 1996; Rohde 1997; Gastin and Oddoux 2001], *alternating linear* [Löding and Thomas 2000], or *linear weak alternating* [Merz and Sezgin 2003; Hammer et al. 2005] automata. Throughout this work, the main constructions used in the transformation of automata and their runs are given full correctness proofs by systematically using a basic toolset of simple results on the properties of runs of generalized alternating automata. Also this toolset is laid out in Ch. 2.

Chapter 3 reviews a basic translation procedure from LTL into self-loop alternating automata. This procedure, which is based on the application of simple translation rules for joining automata built for simple LTL formulas incrementally into more complex automata, is closely related to the construction of Gastin and Oddoux [2001] and satisfies the best known upper bounds for the sizes of components in automata corresponding to LTL formulas. Similarly to related constructions, however, this procedure needs exponential space in the length of the formula in the worst case due to the explicit representation of transitions. This chapter also reviews the result of Rohde [1997] and Löding and Thomas [2000] on the expressive equivalence of LTL and subclasses of alternating automata by presenting a reverse translation from self-loop alternating automata into LTL formulas and analyzing its complexity.

Chapter 4 introduces constructions for translating self-loop alternating ω-automata into nondeterministic ω-automata, again generalizing results presented by Gastin and Oddoux [2001]. Unlike alternating ω-automata in general, self-loop alternating ω-automata with generalized acceptance can be translated into nondeterministic automata using a construction that very closely resembles the classic *subset construction* of Rabin and Scott [1959] for determinizing automata on finite words. In general, however, the nondeterministic ω-automaton may have to use a more complex form of the generalized acceptance condition. This is nevertheless not necessary for a class of alternating automata that have uniform *acceptance synchronized runs* (a new concept introduced in Ch. 4) on all words that they recognize; in particular, all automata built from LTL formulas using the translation procedure from Ch. 3 have this property. Moreover, the nondeterminization construction can be optimized further for these automata by adapting syntactic techniques known from direct translation procedures between LTL and automata [Daniele et al. 1999; Giannakopoulou and Lerda 2002]. In some cases, it is possible to avoid the application of a subset construction entirely if the formula to be translated into an automaton belongs to a special syntactic subclass of LTL which translates directly into alternating automata that support simple completion into nondeterministic automata. This subclass, previously mentioned in the context of symbolic translation algorithms by Schneider [1999], is shown to be closely related also to the syntactic subclass LTL^{det} introduced by Maidl [2000a]. The satisfiability problem of formulas in this subclass of LTL is shown to be **NP**-complete.

Chapter 5 studies the optimization of the basic translation rules presented in Ch. 3 by making use of language containment relationships between self-loop alternating automata. These relationships can also be used to design refined translation rules that aim to simplify the transition structure of automata built with those basic rules which are the main cause of the exponential worst-case space requirements in the basic translation procedure—however, sometimes with a penalty on the number of states in the constructed automata. Some of the refined rules nevertheless prove to be applicable universally as replacements of the basic translation rules without a need to apply computationally expensive tests for language containment relationships. The automata built using the refined rules can still be translated into nondeterministic automata without changing the form of generalized acceptance by applying optimized constructions from Ch. 4. Furthermore, the refined rules also lead to an extension of the syntactic subclass of LTL which translates into automata that can be completed into nondeterministic automata without applying a general nondeterminization construction. The satisfiability problem for this strictly more expressive subclass remains **NP**-complete.

Chapter 6 focuses on the simplification of self-loop alternating automata by making use of language containment tests to remove transitions from the automata. The application of the presented simplification techniques and the refined translation rules from Ch. 5 is illustrated with examples, which include a comparison against the translation procedure proposed by Gastin and Oddoux [2001]. The optimization techniques discussed in Ch. 5 and Ch. 6 are put together in Ch. 7 into a high-level refined translation procedure from LTL into self-loop alternating automata.

Chapter 8 presents a new generalized version of the classic *nested depth-first search* algorithm of Courcoubetis et al. [1991, 1992] for checking the emptiness of nondeterministic automata with generalized transition-based acceptance. The new algorithm improves the search algorithm's worst-case resource requirements, in particular, by reducing the number of additional bits that need to be stored with every visited state (in a simple hash table based implementation) from linear to logarithmic in the number of generalized acceptance conditions.

Finally, Ch. 9 concludes the work by highlighting the main results and discussing directions for further work.

2 DEFINITIONS AND BASIC RESULTS

2.1 MATHEMATICAL CONCEPTS AND NOTATION

We assume basic knowledge on sets, ordered tuples, graphs, trees, relations and functions (mappings), and the principle of mathematical induction. We also refer to basic concepts of computability theory (O-notation, decision problems, deterministic and nondeterministic decision procedures, complexity classes **NP** and **PSPACE**, hardness and completeness for complexity classes) without presenting their formal definitions; see any textbook on computability theory (for example, [Papadimitriou 1994]) for details.

We shall work with the set of natural numbers $\mathbb{N} \stackrel{def}{=} \{0, 1, 2, \ldots\}$ extended with an element $\omega \notin \mathbb{N}$. For notational simplicity, we shall not make use of the formal theory of ordinals in this presentation; instead, we extend the standard total ordering $< \subseteq \mathbb{N} \times \mathbb{N}$ of the natural numbers into $\big(\mathbb{N} \cup \{\omega\}\big) \times \big(\mathbb{N} \cup \{\omega\}\big)$ by defining ω to be an element that satisfies $\omega \not< \omega$ and $n < \omega$ for all $n \in \mathbb{N}$. Comparison between elements of $\mathbb{N} \cup \{\omega\}$ will often be denoted also by the operators $=$, $\leq$, $\geq$ and $>$ with their usual semantics. Furthermore, we also extend addition on the natural numbers to $(\mathbb{N} \cup \{\omega\}) \times (\mathbb{N} \cup \{\omega\})$ by defining $\omega + n \stackrel{def}{=} n + \omega \stackrel{def}{=} \omega + \omega \stackrel{def}{=} \omega$.

Let X and Y be sets. The sets X and Y are *equipollent* if and only if (iff) there exists a bijective mapping $f : X \to Y$. If there exists a natural number $n \in \mathbb{N}$ such that X is equipollent to the (possibly empty) set $\{1, 2, \ldots, n\} \subseteq \mathbb{N}$, we say that X is a *finite* set of *size* n (denoted by $|X| = n$). If X is equipollent to the set of natural numbers $\mathbb{N}$, we say that X is *(countably) infinite* and define $|X| = \omega$. The *powerset* of X (denoted by 2^X) is defined as the set of all subsets of X, i.e., $2^X \stackrel{def}{=} \{Y \mid Y \subseteq X\}$.

If X is a subset of another set Y, we denote by $Y \setminus X$ the *complement of X with respect to Y* (i.e., $Y \setminus X \stackrel{def}{=} \{x \in Y \mid x \notin X\}$). When the set Y is clear from the context, we often denote the complement of X by the shorthand notation $\overline{X}$.

2.1.1 Sequences

Let X be a nonempty set. A *sequence* (called occasionally also a *word* in further discussion) x over X is a mapping $x : I \to X$, from an *index set* $I \stackrel{def}{=} \big\{n \in \mathbb{N} \big| n < m \text{ for some } m \in \mathbb{N} \cup \{\omega\}\big\}$ to X. For all $i \in I$, $x(i)$ is called the $(i+1)^{th}$ *element of* x. We may also describe x by "listing its elements" as $x = (x_i)_{i \in I} = (x_0, x_1, x_2, \ldots)$ (more simply, $x = x_0 x_1 x_2 \ldots$) where $x_i \stackrel{def}{=} x(i)$ for all $i \in I$. We call $|x| \stackrel{def}{=} |I|$ the *length* of the sequence. For all $n \in \mathbb{N} \cup \{\omega\}$, we denote the class of all sequences over X of length n by X^n. The unique sequence in X^0 is called the *empty sequence* over X and is denoted by ε_X. Each element of X can be treated as a sequence by applying the obvious isomorphism between X and X^1. The set $X^* \stackrel{def}{=} \bigcup_{0 \leq i < \omega} X^i$ is the set of all finite sequences over X; X^ω denotes the set of all infinite sequences.

Sequences can be used to define other sequences. Let $x : I \to X$ be a sequence over X, let $i \in I$, and let $0 \leq j \leq |I|$. The sequence $x' : I' \to X$, where $I' \stackrel{def}{=} \{n \in \mathbb{N} \mid i \leq n + i < j\}$ and $x'(k) \stackrel{def}{=} x(k+i)$ for all $k \in I'$, is called a *subsequence* (alternatively, a *subword*) of x and denoted by $x^{[i,j)}$. If $i = 0$, then x' is a *prefix* of x; if $j = |I|$, then x' is called a *suffix*. In this case we usually refer to $x^{[i,j)}$ using the simpler notation x^i. Clearly, the suffix x^i is infinite iff x is infinite. Two subsequences $x_1 = x^{[i_1,j_1)}$ and $x_2 = x^{[i_2,j_2)}$ of a sequence x ($0 \leq i_1, i_2 < |x|$, $0 \leq j_1, j_2 \leq |x|$) are *syntactically identical* (denoted $x_1 = x_2$) iff $|x_1| = |x_2|$ holds, and $x_1(i) = x_2(i)$ holds for all $0 \leq i < |x_1|$; otherwise they are *syntactically distinct* ($x_1 \neq x_2$).

If $x_1 : I_1 \to X_1$ and $x_2 : I_2 \to X_2$ are two sequences with $|x_1| < \omega$, the *concatenation* of x_1 and x_2 (denoted x_1x_2) is the sequence $x : \big\{n \in \mathbb{N} \,\big|\, n < |I_1| + |I_2|\big\} \to X_1 \cup X_2$ defined by

$$x(i) \stackrel{def}{=} \begin{cases} x_1(i) & \text{if } 0 \leq i < |I_1| \\ x_2(i - |I_1|) & \text{if } |I_1| \leq i < |I_1| + |I_2| \end{cases}$$

Because concatenation is an associative operation, i.e., because $(xy)z = x(yz)$ holds for all sequences x, y and z ($|x| < \omega$, $|y| < \omega$), we usually write concatenations of sequences without parentheses.

2.1.2 ω-Regular Expressions

Let X be a nonempty set (called the *alphabet*). We shall often describe subsets of X^ω by means of *ω-regular expressions* over X. The set of ω-regular expressions over X is the smallest set of finite sequences built from elements of X and the symbols (,), $\cup$, *, $^\omega$ (not included in X) such that the set is closed under finite application of the following syntactic rules (formally defined using concatenation of sequences; in the definition of the rules, we also make use of an auxiliary set of *r-expressions* over X):

- Each element of X is an r-expression.
- If α and β are r-expressions, then $(\alpha \cup \beta)$, $(\alpha\beta)$ and α^* are r-expressions.
- If α is an r-expression and β is an ω-regular expression, then α^ω and $(\alpha\beta)$ are ω-regular expressions.
- If α and β are ω-regular expressions, then $(\alpha \cup \beta)$ is an ω-regular expression.

Each r-expression (ω-regular expression) α defines a set of finite (resp. infinite) nonempty words over X. We denote the set of words defined by the r-expression or ω-regular expression α by $\mathcal{L}(\alpha)$ and call this set the *language* of α. Formally, $\mathcal{L}(\alpha)$ is defined for r-expressions and ω-regular expressions as follows:

- $\mathcal{L}(\alpha) \stackrel{def}{=} \big\{x : \{0\} \to X \,\big|\, x(0) = \alpha\big\}$ for all $\alpha \in X$ (i.e., the singleton set containing the unique sequence of length 1 with $\alpha \in X$ as its first element);
- $\mathcal{L}\big((\alpha \cup \beta)\big) \stackrel{def}{=} \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$, where α and β are either both r-expressions or both ω-regular expressions (the sequences that belong to either or both of $\mathcal{L}(\alpha)$ and $\mathcal{L}(\beta)$);

- $\mathcal{L}\big((\alpha\beta)\big) \overset{def}{=} \big\{xy \,\big|\, x \in \mathcal{L}(\alpha),\, y \in \mathcal{L}(\beta)\big\}$ for any r-expression α and any r-expression or ω-regular expression β (the sequences formed by concatenating a sequence from $\mathcal{L}(\beta)$ to a sequence in $\mathcal{L}(\alpha)$);

- $\mathcal{L}(\alpha^*) \overset{def}{=} \{\varepsilon_X\} \cup \bigcup_{1\leq i<\omega} \big\{x_1x_2\ldots x_i \,\big|\, x_j \in \mathcal{L}(\alpha) \text{ for all } 1 \leq j \leq i\big\}$ for any r-expression α (the set of sequences obtained by finite concatenations of zero or more sequences in $\mathcal{L}(\alpha)$)

- $\mathcal{L}(\alpha^\omega) \overset{def}{=} \big\{x_1x_2x_3\ldots \,\big|\, x_i \in \mathcal{L}(\alpha) \setminus \{\varepsilon_X\} \text{ for all } 1 \leq i < \omega\big\}$ for any r-expression α (the set of infinite sequences obtained by concatenating nonempty sequences in the language of α).

Whenever the language of an ω-regular expression α is a singleton set, it is conventional to identify the ω-regular expression with the unique word in its language. In such cases we shall simply speak of the *word* α instead of "the unique word in the language of α". Similarly, we can also identify a nonempty finite word $w : I \to X$ $(0 < |I| < \omega)$ with an r-expression and write w^* and w^ω to denote the languages obtained by finite (resp. infinite) concatenations of the finite word w. In addition, we simplify the general notation by omitting parentheses from r-expressions and ω-regular expressions whenever possible by fixing the precedence of $\cup$, *, $^\omega$ and concatenation such that * and $^\omega$ have precedence over concatenation, which has precedence over $\cup$.

Example 2.1.1 The ω-regular expression a^ω represents the infinite word formed by repeating the symbol a indefinitely, the ω-regular expression $a^\omega \cup b^*c^\omega$ represents all infinite words formed either from an infinite sequence of a's, or a (possibly empty) finite sequence of b's followed by an infinite sequence of c's, and the ω-regular expression $(a \cup b \cup c)^*(ab^*)^\omega$ represents the language of infinite words built from the letters a, b and c such that each word in the language contains infinitely many a's but only finitely many c's.

■

2.2 PROPOSITIONAL LINEAR TIME TEMPORAL LOGIC

As seen in Ex. 2.1.1, ω-regular expressions provide a means for specifying infinite sequences. However, the basic operations for building ω-regular expressions from simpler expressions are not always very convenient for defining languages in practice. For example, even though the languages definable using ω-regular expressions over a given alphabet are closed under the Boolean operations (union, intersection and complementation with respect to the language of all infinite words over the given alphabet) [Büchi 1962; McNaughton 1966], finding an ω-regular expression for a given Boolean combination of languages defined by simpler expressions is often difficult in practice (especially for complementation). This difficulty of combining simple expressions into more complex ones is clearly an undesirable property for a language intended for specifying infinite sequences. Popular specification languages are therefore usually based on alternative formalisms. In particular, languages based on formal logic support expressing Boolean combinations of specifications directly by using the corresponding operations in the logic. Among the best-known such logics suggested for reasoning about the behavior of nonterminating systems is the *propositional linear time temporal logic* (LTL) proposed by Pnueli [Pnueli 1977; Gabbay et al. 1980; Pnueli 1981]. Although this logic formally captures only a strict subset of the sequences specifiable using ω-regular expressions (see, for example, [Thomas 1990]), the logic nevertheless covers a class of specifications that is expressive enough for many actual verification tasks [Manna and Pnueli 1992]. In this section, we shall review the syntax and semantics of this logic.

2.2.1 Syntax

Let AP be a countable set of atomic propositions. The set $LTL(AP)$ of propositional linear time temporal logic formulas over the atomic propositions AP is the smallest set of finite sequences built from elements of AP, parentheses "(" and ")", the symbol $\top$, *propositional connectives* (or *operators*) $\neg$, $\vee$, and *temporal connectives* (operators) X and $\mathsf{U_s}$ such that $LTL(AP)$ includes $\{\top\} \cup AP$ as a subset (where we assume that AP does not include any of the symbols listed above) and is closed under the finite application of the syntactic rule

$$\text{If } \varphi, \psi \in LTL(AP), \text{ then } \neg\varphi, (\varphi \vee \psi), \mathsf{X}\varphi, (\varphi \,\mathsf{U_s}\, \psi) \in LTL(AP).$$

A *subformula* of a formula $\varphi \in LTL(AP)$ is a subsequence of φ that belongs to $LTL(AP)$. The collection of all syntactically distinct subformulas of φ is denoted by $\mathrm{Sub}(\varphi)$. It is easy to check (by induction on the length of φ) that $|\mathrm{Sub}(\varphi)| \leq |\varphi|$ holds for all $\varphi \in LTL(AP)$.

The formula $\varphi \in LTL(AP)$ is called a *literal* iff $\varphi = p$ or $\varphi = \neg p$ holds for some atomic proposition $p \in AP$; literals and the symbol $\top$ are called *atomic* formulas. If $\varphi \notin \{\top\} \cup AP$ holds (i.e., $\varphi = \circ\varphi_1$ for $\circ \in \{\neg, \mathsf{X}\}$, or $\varphi = (\varphi_1 \circ \varphi_2)$, where $\circ \in \{\vee, \mathsf{U_s}\}$, and $\varphi_1, \varphi_2 \in LTL(AP)$), φ is called a *compound formula* with *main connective* $\circ$, and φ_1 and φ_2 are the *top-level subformulas* of φ. The *arity* of a compound formula and its main connective is the number of top-level subformulas in the formula; formulas (connectives) of arity 1 and 2 are called *unary* and *binary* formulas (connectives), respectively.

A formula $\varphi \in LTL(AP)$ that does not contain any temporal connectives is called a *propositional* (or *Boolean*) formula, otherwise it is a *temporal* formula; the set of all propositional formulas over the atomic propositions AP is denoted by $PL(AP)$. The formula φ is called a *pure* temporal formula iff $\varphi = \mathsf{X}\varphi_1$ or $\varphi = (\varphi_1 \,\mathsf{U_s}\, \varphi_2)$ holds for some $\varphi_1, \varphi_2 \in LTL(AP)$. We define the set $\mathrm{Temp}(\varphi)$ as the maximal subset of $\mathrm{Sub}(\varphi)$ which contains only pure temporal formulas.

2.2.2 Semantics

Basic operators

Linear time temporal logic formulas are interpreted over infinite sequences of sets of atomic propositions chosen from AP, i.e., elements of the powerset 2^{AP} of AP. The semantics of linear time temporal logic in an infinite sequence $w \in (2^{AP})^\omega$ of subsets of AP is defined inductively using a binary relation $\models$ as follows:

- $w \models \top$.
- If $p \in AP$, then $w \models p$ iff $p \in w(0)$.
- $w \models \neg\varphi$ iff $w \models \varphi$ does not hold (denoted also by $w \not\models \varphi$).
- $w \models (\varphi \vee \psi)$ iff $w \models \varphi$ or $w \models \psi$.
- $w \models \mathsf{X}\varphi$ iff $w^1 \models \varphi$. [Next time]
- $w \models (\varphi \,\mathsf{U_s}\, \psi)$ iff there exists an index $0 \leq i < \omega$ such that $w^i \models \psi$ holds, and $w^j \models \varphi$ holds for all $0 \leq j < i$. [Strong Until]

We say that $w \in (2^{AP})^\omega$ *satisfies* (alternatively, is a *model* of) the formula $\varphi \in LTL(AP)$ iff $w \models \varphi$ holds. The set $\mathcal{L}(\varphi) \overset{def}{=} \{w \in (2^{AP})^\omega \,|\, w \models \varphi\}$ of all models of φ is called the *language* of φ. The formula φ is *satisfiable* if $\mathcal{L}(\varphi) \neq \emptyset$ holds and *unsatisfiable* otherwise. The formula φ is *valid* iff $\neg\varphi$ is unsatisfiable. For all formulas $\varphi_1, \varphi_2 \in LTL(AP)$, it is clear from the definition of the semantics that $\mathcal{L}\big((\varphi_1 \vee \varphi_2)\big) = \mathcal{L}(\varphi_1) \cup \mathcal{L}(\varphi_2)$, and the complement $\overline{\mathcal{L}(\varphi_1)} \overset{def}{=} (2^{AP})^\omega \setminus \mathcal{L}(\varphi_1)$ of the language of φ_1 with respect to $(2^{AP})^\omega$ equals $\mathcal{L}(\neg\varphi_1)$. For a pair of formulas $\varphi, \psi \in LTL(AP)$, we write $\varphi \equiv \psi$ as a shorthand for $\mathcal{L}(\varphi) = \mathcal{L}(\psi)$; in this case we say that φ and ψ are *logically equivalent*. Clearly, two syntactically identical LTL formulas are always logically equivalent, but the converse does not hold in general (for example, $\varphi \equiv \neg\neg\varphi$ holds for all formulas $\varphi \in LTL(AP)$, but $\varphi \neq \neg\neg\varphi$). If $\varphi \in LTL(AP)$ is an LTL formula with a subformula $\psi \in \mathrm{Sub}(\varphi)$, then it is easy to check from the semantics of propositional linear time temporal logic that $\varphi' \equiv \varphi$ holds for the formula $\varphi' \in LTL(AP)$ obtained from φ by substituting a formula $\psi' \in LTL(AP)$ for any occurrence of the subformula ψ in φ whenever $\psi' \equiv \psi$ holds.

If $\varphi \in PL(AP)$, we project the satisfaction relation from infinite sequences in $(2^{AP})^\omega$ to subsets of AP and use the traditional notation $\sigma \models \varphi$ ($\sigma \subseteq AP$) for propositional satisfiability. (Formally, using the above definition, $\sigma \models \varphi$ is equivalent to the statement that $w \models \varphi$ holds for all $w \in (2^{AP})^\omega$ with $w(0) = \sigma$.)

Derived operators

The set of linear time temporal logic formulas is often extended by introducing derived constants or connectives expressible in terms of the basic constants and connectives $\top$ ("true"), $\neg$ (negation), $\vee$ (disjunction), X (Next Time) and $\mathsf{U_s}$ (Strong Until). The derived connectives allow more flexible expression of LTL properties without changing the expressiveness of the logic. Standard extensions include the Boolean constant $\bot$ ("false": $\bot \overset{def}{\equiv} \neg\top$), the propositional connectives $\wedge$ (conjunction: $(\varphi_1 \wedge \varphi_2) \overset{def}{\equiv} \neg(\neg\varphi_1 \vee \neg\varphi_2)$), $\rightarrow$ (implication: $(\varphi_1 \rightarrow \varphi_2) \overset{def}{\equiv} (\neg\varphi_1 \vee \varphi_2)$), $\leftrightarrow$ (equivalence[1]: $(\varphi_1 \leftrightarrow \varphi_2) \overset{def}{\equiv} \big((\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)\big)$) and $\oplus$ (exclusive disjunction: $(\varphi_1 \oplus \varphi_2) \overset{def}{\equiv} \neg(\varphi_1 \leftrightarrow \varphi_2)$) as well as temporal connectives such as

[1] Although $\equiv$ and $\leftrightarrow$ both capture logical equivalence between formulas, we do not consider the operator $\equiv$ to be part of the (extended) syntax of the logic. This operator will be mainly used for separating different steps in derivations of logically equivalent LTL formulas.

- F: $w \models \mathsf{F}\varphi$ iff $w \models (\top \,\mathsf{U_s}\, \varphi)$. [Finally]
- G: $w \models \mathsf{G}\varphi$ iff $w \models \neg\mathsf{F}\neg\varphi$. [Globally]
- $\mathsf{U_w}$: $w \models (\varphi \,\mathsf{U_w}\, \psi)$ iff $w \models \big(\mathsf{G}\varphi \vee (\varphi \,\mathsf{U_s}\, \psi)\big)$. [Weak Until]
- $\mathsf{R_w}$: $w \models (\varphi \,\mathsf{R_w}\, \psi)$ iff $w \models \neg(\neg\varphi \,\mathsf{U_s}\, \neg\psi)$, equivalently, iff $w \models \big(\psi \,\mathsf{U_w}\, (\varphi \wedge \psi)\big)$. [Weak Release]
- $\mathsf{R_s}$: $w \models (\varphi \,\mathsf{R_s}\, \psi)$ iff $w \models \neg(\neg\varphi \,\mathsf{U_w}\, \neg\psi)$, equivalently, iff $w \models \big(\psi \,\mathsf{U_s}\, (\varphi \wedge \psi)\big)$. [Strong Release]

The sets of propositional and temporal formulas are extended in the obvious way. We also extend the satisfaction relation $\models$ to disjunctions and conjunctions over sets of LTL formulas in the traditional way: for any finite subset $\Phi \subseteq LTL(AP)$ and any $w \in (2^{AP})^\omega$, $w \models \bigvee_{\varphi\in\Phi} \varphi$ ($w \models \bigwedge_{\varphi\in\Phi} \varphi$) holds iff $w \models \varphi$ holds for some (for all) $\varphi \in \Phi$. (By convention, $w \not\models \bigvee_{\varphi\in\emptyset} \varphi$ and $w \models \bigwedge_{\varphi\in\emptyset} \varphi$ hold for all $w \in (2^{AP})^\omega$.) For convenience, we shall occasionally abuse the notation $\bigvee_{\varphi\in\Phi} \varphi$ and $\bigwedge_{\varphi\in\Phi} \varphi$ to denote any (arbitrarily parenthesized) LTL formula formed by joining the elements in the set of formulas $\Phi \subseteq LTL(AP)$ in some order with either the $\vee$ or the $\wedge$ connective, respectively. (By the commutativity and associativity of these connectives, all such formulas are logically equivalent.) For example, this notation can be used to define *conjunctive* and *disjunctive normal forms* of propositional formulas as formulas of the form $\bigwedge_{1\le i\le n} \bigvee_{1\le j\le m_i} \ell_{i,j}$ and $\bigvee_{1\le i\le n} \bigwedge_{1\le j\le m_i} \ell_{i,j}$, respectively, where $\ell_{i,j}$ is a literal (i.e., $\ell_{i,j} \in \{p, \neg p\}$ for some $p \in AP$) for all $1 \le i \le n$ and $1 \le j \le m_i$ ($0 \le n < \omega$, $0 \le m_i < \omega$ for all $1 \le i \le n$).

In this work, we assume all LTL formulas to be written using atomic propositions, Boolean constants $\top$ and $\bot$, and the (extended) set of connectives $\{\neg, \vee, \wedge, \mathsf{X}, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$. All other connectives are assumed to be substituted with their definitions. The subscripts s and w used in the U and R connectives denote the *strength* of the connectives; a formula having one of these connectives as its main connective is called a *strong* (s) or *weak* (w) *temporal eventuality*, respectively. The subscripts will sometimes be omitted if the strength of a connective is not relevant in the context.

The models of the temporal eventualities are infinite sequences over 2^{AP} that have an infinite suffix satisfying a designated top-level subformula (or both top-level subformulas) of the eventuality. The strong temporal eventualities ($\mathsf{U_s}$ and $\mathsf{R_s}$) require the existence of such a suffix unconditionally; their weak variants relax this requirement by permitting models in which another top-level subformula (determined by the type of the connective) holds throughout the entire sequence.[2] In our notation, $\mathsf{U_s}$ and $\mathsf{R_w}$ correspond to the traditional Until and Release connectives commonly used in the literature (see, for example, [Clarke et al. 1999]).

2.2.3 Positive Normal Form

We shall present most of our constructions involving LTL formulas using a syntactically restricted subset of LTL formulas in which the use of negation

[2]For this reason, weak temporal eventualities are sometimes referred to as *invariance* properties in the literature.

Table 2.1: LTL operators and their duals

$\circ$	$\vee$	$\wedge$	$\mathsf{U_s}$	$\mathsf{U_w}$	$\mathsf{R_s}$	$\mathsf{R_w}$
$\tilde{\circ}$	$\wedge$	$\vee$	$\mathsf{R_w}$	$\mathsf{R_s}$	$\mathsf{U_w}$	$\mathsf{U_s}$

is allowed only immediately before atomic propositions. Formally, we define this set $LTL^{\mathrm{PNF}}(AP)$ of LTL formulas (over a set AP of atomic propositions) in *positive normal form* as

$$LTL^{\mathrm{PNF}}(AP) \stackrel{def}{=} \big\{\varphi \in LTL(AP) \,\big|\, \text{for all } 0 \le i < |\varphi| - 1 : \text{ if } \varphi(i) = \neg, \text{ then } \varphi(i+1) \in AP\big\}.$$

The restriction to $LTL^{\mathrm{PNF}}(AP)$ does not reduce the expressive power of the logic when using the extended set of LTL operators fixed above: any formula $\varphi \in LTL(AP)$ can be written as a logically equivalent LTL formula $[\varphi]^{\mathrm{PNF}}$ in positive normal form defined recursively as follows:

$$\begin{aligned}
[\varphi]^{\mathrm{PNF}} &\stackrel{def}{=} \varphi \qquad \text{if } \varphi \text{ is an atomic formula}\\
[\neg\top]^{\mathrm{PNF}} &\stackrel{def}{=} \bot\\
[\neg\bot]^{\mathrm{PNF}} &\stackrel{def}{=} \top\\
[\neg\neg\varphi_1]^{\mathrm{PNF}} &\stackrel{def}{=} [\varphi_1]^{\mathrm{PNF}}\\
[\mathsf{X}\varphi_1]^{\mathrm{PNF}} &\stackrel{def}{=} \mathsf{X}[\varphi_1]^{\mathrm{PNF}}\\
[\neg\mathsf{X}\varphi_1]^{\mathrm{PNF}} &\stackrel{def}{=} \mathsf{X}[\neg\varphi_1]^{\mathrm{PNF}}\\
\big[(\varphi_1 \circ \varphi_2)\big]^{\mathrm{PNF}} &\stackrel{def}{=} \big([\varphi_1]^{\mathrm{PNF}} \circ [\varphi_2]^{\mathrm{PNF}}\big)\\
\big[\neg(\varphi_1 \circ \varphi_2)\big]^{\mathrm{PNF}} &\stackrel{def}{=} \big([\neg\varphi_1]^{\mathrm{PNF}} \,\tilde{\circ}\, [\neg\varphi_2]^{\mathrm{PNF}}\big)
\end{aligned}$$

where $\varphi_1, \varphi_2 \in LTL(AP)$ are the top-level subformulas of φ if φ is a compound formula, $\circ \in \{\vee, \wedge, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$, and $\tilde{\circ}$ is the *dual operator* of $\circ$ defined as shown in Table 2.1.

It is easy to check (by induction on $|\varphi|$) that $[\varphi]^{\mathrm{PNF}}$ is in positive normal form. Similarly, $[\varphi]^{\mathrm{PNF}} \equiv \varphi$ holds since each case in the definition of $[\varphi]^{\mathrm{PNF}}$ is based on some LTL identity (in particular, the "generalized" De Morgan law $\neg(\varphi_1 \circ \varphi_2) \equiv (\neg\varphi_1 \,\tilde{\circ}\, \neg\varphi_2)$ holds for all $\varphi_1, \varphi_2 \in LTL(AP)$ and $\circ \in \{\vee, \wedge, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$). Furthermore, it is easy to see that $\big|[\varphi^{\mathrm{PNF}}]\big| < 2 \cdot |\varphi|$ holds, and thus $\big|\mathrm{Sub}\big([\varphi]^{\mathrm{PNF}}\big)\big| \le \big|[\varphi^{\mathrm{PNF}}]\big| < 2 \cdot |\varphi|$. Similarly, because the number of temporal operators in $[\varphi]^{\mathrm{PNF}}$ is always equal to their number in φ (as is again easily checked from the recursive definition), $[\varphi]^{\mathrm{PNF}}$ has at most twice as many syntactically distinct pure temporal subformulas as φ (each pure temporal subformula of $[\varphi]^{\mathrm{PNF}}$ either equals $[\psi]^{\mathrm{PNF}}$ or $[\neg\psi]^{\mathrm{PNF}}$ for some $\psi \in \mathrm{Temp}(\varphi)$).

Example 2.2.1 We find the positive normal form of the LTL formula

$$\varphi \stackrel{def}{=} \neg\Big(\big((\neg p_1 \,\mathsf{R_w}\, p_2) \wedge \neg(\bot \,\mathsf{R_w}\, \neg p_1)\big) \wedge \mathsf{X}p_2\Big) \in LTL\big(\{p_1, p_2\}\big).$$

Applying the recursive definition, we get

$$
\begin{aligned}
[\varphi]^{\mathrm{PNF}} &= \big[\neg\big(((\neg p_1 \,\mathsf{R_w}\, p_2) \wedge \neg(\bot \,\mathsf{R_w}\, \neg p_1)) \wedge \mathsf{X}p_2\big)\big]^{\mathrm{PNF}} \\
&= \big(\big[\neg((\neg p_1 \,\mathsf{R_w}\, p_2) \wedge \neg(\bot \,\mathsf{R_w}\, \neg p_1))\big]^{\mathrm{PNF}} \vee [\neg \mathsf{X} p_2]^{\mathrm{PNF}}\big) \\
&= \big(\big([\neg(\neg p_1 \,\mathsf{R_w}\, p_2)]^{\mathrm{PNF}} \vee [\neg\neg(\bot \,\mathsf{R_w}\, \neg p_1)]^{\mathrm{PNF}}\big) \vee \mathsf{X}[\neg p_2]^{\mathrm{PNF}}\big) \\
&= \big(\big(([\neg\neg p_1]^{\mathrm{PNF}} \,\mathsf{U_s}\, [\neg p_2]^{\mathrm{PNF}}) \vee [(\bot \,\mathsf{R_w}\, \neg p_1)]^{\mathrm{PNF}}\big) \vee \mathsf{X}\neg p_2\big) \\
&= \big(\big(([p_1]^{\mathrm{PNF}} \,\mathsf{U_s}\, \neg p_2) \vee ([\bot]^{\mathrm{PNF}} \,\mathsf{R_w}\, [\neg p_1]^{\mathrm{PNF}})\big) \vee \mathsf{X}\neg p_2\big) \\
&= \big(\big((p_1 \,\mathsf{U_s}\, \neg p_2) \vee (\bot \,\mathsf{R_w}\, \neg p_1)\big) \vee \mathsf{X}\neg p_2\big).
\end{aligned}
$$

■

For all formulas $\varphi \in LTL^{\mathrm{PNF}}(AP)$, we define the *node size* of φ (denoted by $\mathrm{NSize}(\varphi)$) and the set of *node subformulas* of φ ($\mathrm{NSub}(\varphi)$) as follows:

$$
\begin{aligned}
\mathrm{NSize}(\varphi) &\stackrel{def}{=} 1 + \begin{cases} 0 & \varphi \text{ atomic} \\ \sum_{\psi \in \{\psi' \in \mathrm{Sub}(\varphi) \mid \psi' \text{ top-level}\}} \mathrm{NSize}(\psi) & \text{otherwise} \end{cases} \\
\mathrm{NSub}(\varphi) &\stackrel{def}{=} \{\varphi\} \cup \begin{cases} \emptyset & \varphi \text{ atomic} \\ \bigcup_{\psi \in \{\psi' \in \mathrm{Sub}(\varphi) \mid \psi' \text{ top-level}\}} \mathrm{NSub}(\psi) & \text{otherwise} \end{cases}
\end{aligned}
$$

Informally, $\mathrm{NSize}(\varphi)$ corresponds to the number of nodes in a nonempty labeled tree (labeled with subformulas of $\varphi \in LTL^{\mathrm{PNF}}(AP)$) such that the root of the tree is labeled with φ itself, and each node labeled with a non-atomic subformula $\psi \in \mathrm{Sub}(\varphi)$ has one or two children labeled with ψ's top-level subformulas. The set $\mathrm{NSub}(\varphi) \subseteq \mathrm{Sub}(\varphi)$ can then be identified as the set of syntactically distinct node labels occurring in the tree (alternatively, the set of nodes in a directed acyclic graph obtained from the tree by merging nodes with identical labels). Because φ is in positive normal form, no internal node of the tree (i.e., a node with at least one child node) is labeled with a formula of the form $\neg\psi$.

Clearly, $\mathrm{NSize}(\varphi) \leq |\varphi|$ holds for all $\varphi \in LTL^{\mathrm{PNF}}(AP)$, and for an arbitrary LTL formula $\varphi \in LTL(AP)$,

$$
\big|\mathrm{NSub}\big([\varphi]^{\mathrm{PNF}}\big)\big| \leq \mathrm{NSize}\big([\varphi]^{\mathrm{PNF}}\big) \leq \big|[\varphi^{\mathrm{PNF}}]\big| < 2 \cdot |\varphi|.
$$

$\mathrm{NSize}(\varphi)$ and $\mathrm{NSub}(\varphi)$ will be useful for analyzing constructions involving LTL formulas in positive normal form.

Example 2.2.2 Figure 2.1 shows the labeled tree representation of the LTL formula

$$
\psi \stackrel{def}{=} \big(\big((p_1 \,\mathsf{U_s}\, \neg p_2) \vee (\bot \,\mathsf{R_w}\, \neg p_1)\big) \vee \mathsf{X}\neg p_2\big) \in LTL^{\mathrm{PNF}}\big(\{p_1, p_2\}\big)
$$

defined in the previous example. From the tree representation it is easy to verify that $\mathrm{NSize}(\psi) = 10$, and $|\mathrm{NSub}(\psi)| = \big|\big\{\psi, \big((p_1 \mathsf{U_s} \neg p_2) \vee (\bot \mathsf{R_w} \neg p_1)\big), (p_1 \mathsf{U_s} \neg p_2), p_1, \neg p_2, (\bot \mathsf{R_w} \neg p_1), \bot, \neg p_1, \mathsf{X}\neg p_2\big\}\big| = 9.$ ■

2.3 ALTERNATING AUTOMATA

As a third formalism for characterizing sets of infinite words we consider finite automata, i.e., finite state machines that are able to recognize words

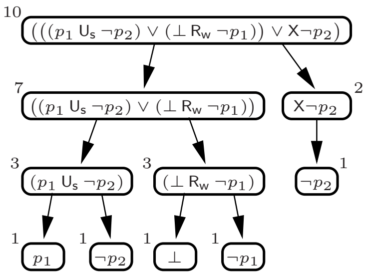


Fig. 2.1: The LTL formula $\big(((p_1 \mathsf{U_s} \neg p_2) \vee (\bot \mathsf{R_w} \neg p_1)\big) \vee \mathsf{X}\neg p_2\big)$ as a labeled tree. The number beside a node corresponds to the value of NSize(ψ) for the LTL formula ψ labeling the node

by reducing a given property of a word into a property of a state sequence generated by the machine when given the word as input. The automaton generates this state sequence by examining the word one symbol at a time, choosing the next state to be generated from a finite (possibly empty) set of alternatives determined by the most recently generated state and the next symbol in the input. The concept of mixing this nondeterministic (existential) choice between several alternatives with universal choice for the next state to be generated (intuitively, choosing multiple "next" states at once) was first proposed for finite automata working on finite words by Kozen [1976] and Chandra and Stockmeyer [1976], who defined the expressively equal notions of *parallel* and *alternating* finite automata, respectively (and later combined their work in [Chandra et al. 1981]). The *Boolean* automata of Brzozowski and Leiss [1980] provide another equivalent way to combine existential and universal behavior in finite automata. This concept of *alternation* was later extended to automata working on infinite words and trees by Miyano and Hayashi [1984a,b] and Muller and Schupp [1985, 1987], respectively. Unlike a nondeterministic automaton, which always generates at most one next state at each step when working on its input, an alternating automaton can at any step generate multiple "next" states at once and spawn copies of itself that work independently on the remaining input. Thus, instead of mapping its input to a single state sequence, an alternating automaton may generate a collection of such sequences: whether the automaton recognizes ("accepts") its input is then determined from the properties of this collection of sequences.

In the case of finite words, alternation allows for a succinct representation of "acceptable" sequences that does not add to the expressiveness of plain nondeterministic automata: for every alternating automaton accepting a set of finite words using n states, there exists a nondeterministic automaton that accepts the same set of words. The nondeterministic automaton may have an exponential number of states in n in the worst case, however, because the minimal *deterministic* automaton simulating an alternating one may have a doubly exponential number of states in n in the worst case [Kozen 1976; Chandra and Stockmeyer 1976; Brzozowski and Leiss 1980; Chandra et al. 1981; Leiss 1981]. An analogous correspondence holds between classes of alternating and nondeterministic automata on infinite words under many notions of acceptance [Miyano and Hayashi 1984a,b; Lindsay 1988; Muller and Schupp 1995]. Together with the expressive power of automata (which can be made to coincide with that of the ω-regular expressions by using an appropriate notion for acceptance [Büchi 1962; McNaughton 1966]) and the suitability of automata for algorithmic analysis in general, the succinctness of alternating automata provides the main motivation for applying them to the specification and automatic verification of properties of infinite sequences. In this section, we review the basic definitions and properties of alternating automata on infinite words and some subclasses of the automata.

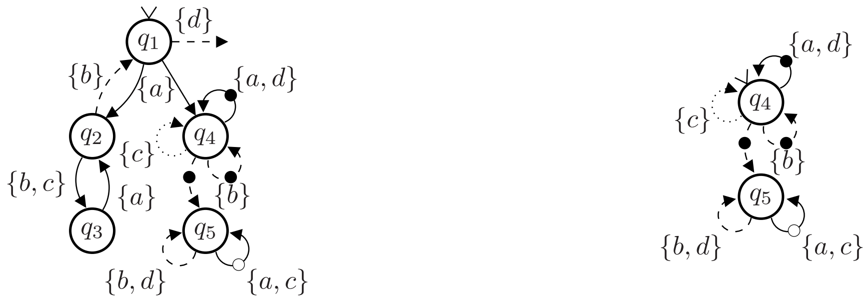
2.3.1 Basic Concepts

The combination of existential and universal choice between states of alternating automata can be captured by encoding the transitions of the automata as arbitrary Boolean functions (formulas) on the states of the automata [Kozen 1976; Chandra and Stockmeyer 1976; Brzozowski and Leiss 1980; Chandra et al. 1981]; however, it is common to restrict the use of negation [Chandra and Stockmeyer 1976] especially when working with infinite inputs [Muller and Schupp 1985, 1987; Vardi 1994]. Although the Boolean representation is convenient for proving many fundamental properties of alternating automata, such as their eligibility to an elegant complementation construction based on syntactic manipulation of the Boolean functions and "dualization" of the notion of acceptance [Muller and Schupp 1985, 1987], the notion of a single transition is not always explicit in the Boolean representation. Such a notion is nevertheless useful, for example, for representing automata graphically as traditional state graphs. As we shall see later in Ch. 6, an explicit representation for the transitions is also convenient for the simplification of the automata. For these reasons, we adopt a definition similar to the one used previously by Gastin and Oddoux [2001]; in this definition, the individual transitions leaving a state correspond to the disjuncts in the disjunctive normal form of a corresponding Boolean function built from the Boolean constants, the states of the automaton and the $\vee$ and the $\wedge$ connectives.

Formally, an *alternating automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$, where Σ is a finite set called the *alphabet*, Q is the finite set of *states*, $q_I \in Q$ is the *initial state*, $\Delta \subseteq Q \times 2^{\Sigma} \times 2^{\mathcal{F}} \times 2^{Q}$ is the *transition relation* and $\mathcal{F}$ is the finite set of *acceptance conditions*.

Individual elements in the transition relation Δ are called *transitions* of the automaton. The components of a transition $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ (for some $q \in Q$, $\Gamma \subseteq \Sigma, F \subseteq \mathcal{F}, Q' \subseteq Q$) are called the *source state*, the *guard*, the *acceptance conditions*, and the *target states* of t, respectively. The transition t is an *initial transition* of $\mathcal{A}$ iff the source state of t is the initial state of the automaton $\mathcal{A}$ ($q = q_I$), a *self-loop* iff it includes its own source state in its target states ($q \in Q'$), and (for an acceptance condition $f \in \mathcal{F}$) an *f-transition* iff it includes the condition f in its acceptance conditions ($f \in F$). A state $q' \in Q$ is an *f-state* iff it is the source state of an f-transition in $\mathcal{A}$.

The well-known class of nondeterministic finite automata arises as a special case of alternating automata in which $|Q'| = 1$ holds for every transition $\langle q, \Gamma, F, Q' \rangle \in \Delta$. In other words, an alternating automaton is *nondeterministic* iff its every transition has exactly one target state. Many questions about properties of nondeterministic automata can be answered using graph-



(a) (b)

Fig. 2.2: (a) The alternating automaton $\mathcal{A} = \big\langle \{a,b,c,d\}, \{q_1,q_2,q_3,q_4,q_5\}, \big\{\langle q_1,\{a\},\emptyset,\{q_2,q_4\}\rangle, \langle q_1,\{d\},\emptyset,\emptyset\rangle, \langle q_2,\{b\},\emptyset,\{q_1\}\rangle, \langle q_2,\{b,c\},\emptyset,\{q_3\}\rangle, \langle q_3,\{a\},\emptyset,\{q_2\}\rangle, \langle q_4,\{a,d\},\{\bullet\},\{q_4\}\rangle, \langle q_4,\{b\},\{\bullet\},\{q_4,q_5\}\rangle, \langle q_4,\{c\},\emptyset,\{q_4\}\rangle, \langle q_5,\{a,c\},\{\circ\},\{q_5\}\rangle, \langle q_5,\{b,d\},\emptyset,\{q_5\}\rangle\big\}, q_1, \{\bullet,\circ\}\big\rangle$;
(b) The subautomaton $\mathcal{A}^{q_4} = \big\langle \{a,b,c,d\}, \{q_4,q_5\}, \big\{\langle q_4,\{a,d\},\{\bullet\},\{q_4\}\rangle, \langle q_4,\{b\},\{\bullet\},\{q_4,q_5\}\rangle, \langle q_4,\{c\},\emptyset,\{q_4\}\rangle, \langle q_5,\{a,c\},\{\circ\},\{q_5\}\rangle, \langle q_5,\{b,d\},\emptyset,\{q_5\}\rangle\big\}, q_4, \{\bullet,\circ\}\big\rangle$

theoretic decision procedures. We shall discuss the construction and analysis of such automata in Ch. 4 and Ch. 8.

Example 2.3.1 We illustrate our conventions for drawing alternating automata in Fig. 2.2 (a). The states of the automata are drawn as circles (with the initial state of the automaton marked by a small arrowhead), and the transitions of the automata are represented by sets of arrows connecting the circles. We occasionally omit the labels of the states if they are not relevant in the context. For each transition $\langle q, \Gamma, F, Q'\rangle \in \Delta$ with a nonempty set of target states (i.e., if $|Q'| = n$ holds for some $1 \leq n < \omega$), we draw n arrows from the state q to each state in Q'. It is also permissible for a transition to have no target states (in which case $Q' = \emptyset$ holds): every such transition (such as the transition with guard $\{d\}$ starting from the state q_I in Fig. 2.2 (a)) is represented with a single arrow connected only to the transition's source state. Arrows associated with the same transition are drawn in the same line style; since the source of each transition is unique, the same line styles can be reused in each state of the automaton without ambiguity as far as the correspondence between arrows and transitions is concerned. Acceptance conditions in F are represented by small shaded circles on the transition arrows; each different shade corresponds to a different acceptance condition. To simplify the figures, we usually place the transition guards near only one of the arrows associated with a particular transition. We nevertheless repeat the acceptance conditions of the transition on each of these arrows. ■

Successors, Paths, Descendants and Subautomata

Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ be an alternating automaton, and let $q \in Q$. A state $q' \in Q$ is a *successor* of q if there exists a transition $\langle q, \Gamma, F, Q'\rangle \in \Delta$ such that $q' \in Q'$. A *path* in $\mathcal{A}$ is a nonempty sequence $x = (q_i)_{0 \leq i < n} \in Q^n$, where $1 \leq n \leq \omega$, and q_i is a successor of q_{i-1} for all $1 \leq i < n$. If $n = 1$, the path is *trivial*; if $n < \omega$, the path is a *finite* path *from* q_0 *to* q_{n-1}; otherwise the path is *infinite*. The *length* of the path is the length of the sequence x, i.e., the number of states in the sequence. The path *visits* the state $q \in Q$ iff $q_i = q$ holds for some $0 \leq i < n$. The path is *simple* iff $q_i \neq q_j$ holds for all

$0 \leq i, j < n$ ($i \neq j$), and it is a *loop* (alternatively, a *cycle*) iff it is a finite nontrivial path with $q_{n-1} = q_0$. The cycle is *simple* iff $(q_0, q_1, \ldots, q_{n-2})$ is a simple path. We reuse the terminology introduced for the transitions of $\mathcal{A}$ and call a loop of length 2 a *self-loop*. (A self-loop transition always defines a path that is a self-loop, but the converse does not hold in the general case.) A state $q' \in Q$ is a *descendant* of $q \in Q$ iff there exists a finite nontrivial path from q to q' in the automaton; in this case we also say that q' is *reachable* from q in $\mathcal{A}$.

Let $t_1 = \langle q_1, \Gamma_1, F_1, Q'_1 \rangle \in \Delta$ and $t_2 = \langle q_2, \Gamma_2, F_2, Q'_2 \rangle \in \Delta$ be two transitions of $\mathcal{A}$. The transition t_2 is *consecutive* to the transition t_1 iff the source state of t_2 is a target state of t_1 (i.e., iff $q_2 \in Q'_1$ holds). A *transition chain* is a (possibly empty) sequence of transitions $(t_i)_{0 \leq i < n} \in \Delta^n$ ($0 \leq n \leq \omega$) such that t_i is consecutive to t_{i-1} for all $1 \leq i < n$. We say that the chain is *maximal* if the chain cannot be concatenated with a transition (in either order) to obtain another chain of transitions. Analogously to paths, the *length* of the chain is the number of transitions (n) in the chain, the chain is *finite* if $n < \omega$ and *infinite* otherwise, the chain *visits* a state $q \in Q$ iff q is the source state of t_i for some $0 \leq i < n$, and it is *simple* iff $t_i \neq t_j$ holds for all $0 \leq i, j < n$ ($i \neq j$).

Example 2.3.2 In Fig. 2.2 (a), the successors of q_1 are the states q_2 and q_4; the descendants of q_1 include also the states q_3 and q_5, because $x_1 \stackrel{def}{=} (q_1, q_2, q_3)$ and $x_2 \stackrel{def}{=} (q_1, q_4, q_5, q_5)$ are finite nontrivial paths (of lengths 3 and 4) from q_1 to q_3 and q_5, respectively. Because x_1 does not visit any of the states q_1, q_2 or q_3 twice, x_1 is simple; this does not hold, however, for the path x_2 that includes a self-loop (q_5, q_5) from q_5 to itself. The path $x_3 \stackrel{def}{=} (q_2, q_3, q_2, q_3, q_2, q_3, \ldots)$ is an infinite path that begins with a cycle (q_2, q_3, q_2) that is not a self-loop. The infinite transition chain $y \stackrel{def}{=} \big(\langle q_1, \{a\}, \emptyset, \{q_2, q_4\}\rangle, \langle q_4, \{c\}, \emptyset, \{q_4\}\rangle, \langle q_4, \{c\}, \emptyset, \{q_4\}\rangle, \langle q_4, \{c\}, \emptyset, \{q_4\}\rangle, \ldots\big)$ visits the states q_1 and q_4. ■

Let $q \in Q$ be a state in the automaton $\mathcal{A}$. The *subautomaton* of $\mathcal{A}$ with initial state q (denoted $\mathcal{A}^q$) is the alternating automaton obtained from $\mathcal{A}$ by changing its initial state to q, removing all states that are different from q but that are not descendants of q from the resulting automaton and restricting the transition relation Δ to the remaining set of states. The subautomaton shares its set of acceptance conditions with the original automaton. We also say that $\mathcal{A}^q$ is *rooted at* the state $q \in Q$. Formally, $\mathcal{A}^q \stackrel{def}{=} \langle \Sigma, Q^q, \Delta^q, q^q_I, \mathcal{F}^q \rangle$, where $Q^q \stackrel{def}{=} \{q\} \cup \{q' \in Q \mid q' \text{ is a descendant of } q\}$, $\Delta^q \stackrel{def}{=} \big\{\langle q', \Gamma, F, Q' \rangle \in \Delta \,\big|\, \{q'\} \cup Q' \subseteq Q^q\big\}$, $q^q_I \stackrel{def}{=} q$, and $\mathcal{F}^q \stackrel{def}{=} \mathcal{F}$. It is easy to see that if $\mathcal{A}^q$ is a subautomaton of $\mathcal{A}$ and $q' \in Q^q$, then $(\mathcal{A}^q)^{q'} = \mathcal{A}^{q'}$, i.e., each subautomaton of $\mathcal{A}^q$ is also a subautomaton of $\mathcal{A}$.

When taking subautomata of alternating automata, we sometimes also project the acceptance conditions of $\mathcal{A}$ into another given set of acceptance conditions $\mathcal{F}'$. Formally, for a set $\mathcal{F}'$, we denote by $\mathcal{A}^{q,\mathcal{F}'}$ the automaton $\langle \Sigma, Q^q, \Delta^{q,\mathcal{F}'}, q^q_I, \mathcal{F}' \rangle$, where Q^q and q^q_I are defined as above, and $\Delta^{q,\mathcal{F}'} \stackrel{def}{=} \big\{\langle q', \Gamma, F \cap \mathcal{F}', Q' \rangle \,\big|\, \langle q', \Gamma, F, Q' \rangle \in \Delta, \{q'\} \cup Q' \subseteq Q^q\big\}$.

Example 2.3.3 Figure 2.2 (b) shows the subautomaton $\mathcal{A}^{q_4}$ obtained from the automaton $\mathcal{A}$ depicted in Fig. 2.2 (a). ■

Runs

A *run* of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ on an infinite word $w \in \Sigma^\omega$ is a directed labeled acyclic graph (DAG) $G = \langle V, E, L \rangle$, where the set of *nodes* V, the set of *(hyper)edges* E and the *labeling function* $L : V \cup E \to Q \cup \Delta$ satisfy the following conditions (for $X \subseteq V \cup E$, we write $L(X)$ as a shorthand for $\{L(x) \mid x \in X\}$):

- The set of nodes V can be partitioned into finite pairwise disjoint *levels* $V_i \subseteq V$ (i.e., $V = \bigcup_{0 \leq i < \omega} V_i$, $V_i \cap V_j \neq \emptyset$ for all $0 \leq i, j < \omega$, $i \neq j$) such that $V_0 = \{v_0\}$ is a singleton, and $E \subseteq \bigcup_{0 \leq i < \omega} (V_i \times 2^{V_{i+1}})$. [Partitioning to levels]
- For all $v \in V$, there exists a unique edge $\langle v, V' \rangle \in E$. [Forward causality]
- For all $v' \in V \setminus V_0$, there exists an edge $\langle v, V' \rangle \in E$ such that $v' \in V'$. [Backward causality]
- $L(v_0) = q_I$, and for all $0 \leq i < \omega$ and $e = \langle v, V' \rangle \in E \cap (V_i \times 2^{V_{i+1}})$, there exists a transition $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ such that $L(e) = t$, $L(v) = q$, $w(i) \in \Gamma$, and $L(V') = Q'$. [Consistency of labeling]

Let $e = \langle v, V' \rangle \in E$. Reusing the terminology defined for alternating automata, we call each element $v' \in V'$ a *successor* of v; source and target nodes of edges, descendants of a node, paths in a run, consecutive edges, and chains and maximal chains of edges are defined in the obvious way.

Graph- vs. tree-based runs. Note that, although every individual level of a run is required to be finite, we do not require the maximum number of nodes in a level of a run to be finitely bounded. Therefore, the above graph-based definition of runs subsumes (but is no more general than) the more common definition of runs as finitely branching labeled trees (see, for example, [Vardi 1995]) in which the partitioning and backward causality properties follow implicitly from the properties of trees. The graph-based definition is convenient for expressing results on runs simply as transformations between runs without a need to introduce additional concepts such as "run DAGs" (as often used in the literature [Isli 1994, 1996; Kupferman and Vardi 1997, 2001]).

Example 2.3.4 Figure 2.3 illustrates the construction of the first few levels of (one possible) run for the alternating automaton in Fig. 2.2 (a) on an input word that begins with the symbols $acabacababdcd$. Again, we represent nodes of the run with circles and edges with (sets) of arrows connecting the nodes; we first draw the node corresponding to the level V_0 of the run (the leftmost node in the figure) and label this node with the initial state q_1 of the automaton. On the first input symbol a, the automaton spawns two copies of itself that then process the next symbol of the input, starting from the states q_2 and q_4, respectively. The spawning of the two copies is represented by drawing arrows from the node labeled with the state q_1 to two new nodes labeled with these states in the figure; these nodes form level V_1 of the run. Nodes on the same level are always drawn horizontally aligned (with their labels shown beside the nodes themselves). Subsequent levels of the run

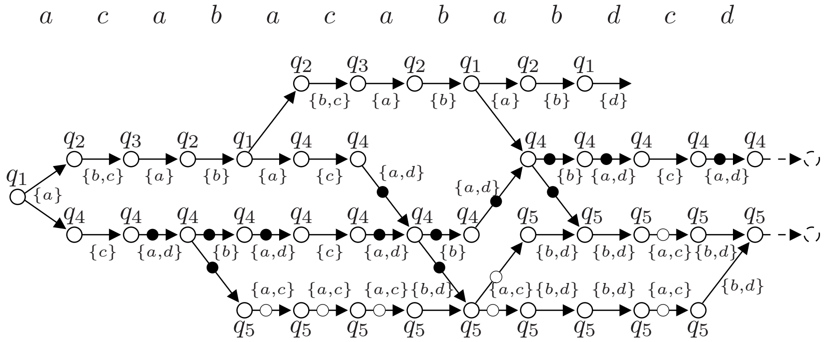


Fig. 2.3: First few levels of a run of the alternating automaton in Fig. 2.2 (a) formed by reading the input $acabacababdcd$

of the automaton are defined in a similar manner so that the labels of the successors of each node always match the target states of some transition that starts from the state labeling the node itself and includes the next input symbol (shown at the top of the figure) in its guard.

The nodes in a level of the run represent the copies of the automaton which are "active" at the corresponding position of the input. As seen in the figure, the number of active copies (of which several may be in the same state of the automaton) can change while the automaton processes its input: the automaton may spawn new copies of itself, some of the copies may "merge together", or a copy of the automaton may "die" by taking a transition with an empty set of target states. Note that the structural properties of a run do not enforce the merging of copies; nor do they enforce any global finite upper bound on the number of copies of a particular subautomaton (or the total number of active copies of the automaton) spawned at any position of the input.

The arrows leaving each node constitute the unique (hyper)edge starting from the node. Formally, each of these edges is labeled with a transition of the automaton. We mark only the guards and the acceptance conditions of these transitions in the figure; the exact label of each edge in the run can nevertheless be determined uniquely from this information together with the labels of the nodes. ■

Infinite Branches and Their Properties

Let $G = \langle V, E, L\rangle$ be a run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ on an infinite word $w \in \Sigma^\omega$. We call each maximal chain of edges in G a *branch* of G; by the partitioning and backward causality, it is easy to see that each branch of G begins with the unique edge $\langle v_0, V_1\rangle \in E$. In particular, we denote the set of infinite branches in G by $\mathcal{B}(G)$ and define, for each infinite branch $\beta = (e_i)_{0\leq i<\omega} \in \mathcal{B}(G)$,

$$\inf(\beta) \stackrel{def}{=} \big\{f \in \mathcal{F} \,\big|\, \forall i \geq 0 : \exists j \geq i : L(e_j) \text{ is an } f\text{-transition of } \mathcal{A}\big\}$$

and

$$\operatorname{fin}(\beta) \stackrel{def}{=} \big\{f \in \mathcal{F} \,\big|\, \exists i \geq 0 : \forall j \geq i : L(e_j) \text{ is an } f\text{-transition of } \mathcal{A}\big\}.$$

The sets $\inf(\beta)$ and $\mathrm{fin}(\beta)$ are called the *infinity set* and the *final set* of β, respectively; $\inf(\beta)$ collects the acceptance conditions occurring in the label of infinitely many edges in the branch β, and $\mathrm{fin}(\beta)$ is the maximal set of conditions that are missing from the labels of only finitely many edges in the branch. It is easy to see that if $\mathrm{fin}(\beta) \neq \emptyset$, then $\inf(\beta) \neq \emptyset$, and furthermore, if $\inf(\beta) \neq \emptyset$, then, for all $f \in \inf(\beta)$, there exists an f-transition $t_f \in \Delta$ such that $L(e_i) = t_f$ holds for infinitely many i, because Δ is finite.

Acceptance Modes and the Language of an Automaton
Let $G = \langle V, E, L\rangle$ be a run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ on an infinite word $w \in \Sigma^\omega$, and let $\mathcal{B}(G)$ be the set of infinite branches in G. A branch $\beta \in \mathcal{B}(G)$ is *inf-accepting* iff $\inf(\beta) = \mathcal{F}$ and *fin-accepting* iff $\mathrm{fin}(\beta) = \emptyset$. The run G is *inf- (fin-)accepting* iff every branch $\beta \in \mathcal{B}(G)$ is inf- (fin-)accepting, respectively. (If all branches of G are finite, then $\mathcal{B}(G) = \emptyset$, and thus G is trivially both inf- and fin-accepting.)

We say that $\mathcal{A}$ *inf-accepts* (*fin-accepts*) $w \in \Sigma^\omega$ iff $\mathcal{A}$ has an inf-accepting (fin-accepting) run on w. We call the set of infinite words accepted by $\mathcal{A}$ in a fixed acceptance mode the *language* of $\mathcal{A}$ and denote it by $\mathcal{L}_{\inf}(\mathcal{A})$ or $\mathcal{L}_{\mathrm{fin}}(\mathcal{A})$, where the acceptance mode is given in the subscript. The automaton $\mathcal{A}$ *inf- or fin-recognizes* a language $\mathcal{L} \subseteq \Sigma^\omega$ iff $\mathcal{L} = \mathcal{L}_{\inf}(\mathcal{A})$ or $\mathcal{L} = \mathcal{L}_{\mathrm{fin}}(\mathcal{A})$, respectively. The automaton is *inf- (fin-)empty* iff it inf- (fin-)recognizes the empty language. We call two automata *inf- (fin-)equivalent* iff they inf- (fin-)recognize the same language.

Transition- vs. state-based acceptance. In the literature, acceptance is often characterized by associating acceptance conditions with states instead of the transitions of an automaton and defining (for an infinite branch β in a run of the automaton) $\inf(\beta)$ ($\mathrm{fin}(\beta)$) to be the union of the acceptance conditions associated with those states in the automaton that occur as labels of the source nodes of infinitely many (resp. all except for finitely many) edges in β. We shall refer to this form of acceptance as *state-based* (inf- or fin-)acceptance to distinguish it from the above notion of *transition-based* acceptance. It is a well-known fact that state-based inf-acceptance can always be reduced to transition-based inf-acceptance without altering the state set of the automaton, but the converse reduction may necessitate duplicating some states of the automaton to preserve its language; the results on nondeterministic automata (see, for example, [Perrin and Pin 2004]) generalize easily to alternating automata with multiple acceptance conditions.

Relation to generalized Büchi acceptance. Our notion of inf-acceptance is equivalent to classic (generalized) *Büchi acceptance* commonly used in the literature (see, for example, the survey article by Thomas [1990]). As a matter of fact, fin-acceptance can be seen merely as a way to rephrase the notion of inf-acceptance, because inf- and fin-acceptance can be easily reduced to each other: an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ fin- (inf-)recognizes the language $\mathcal{L} \subseteq \Sigma^\omega$ iff the automaton obtained from $\mathcal{A}$ by complementing the set of acceptance conditions of every transition of $\mathcal{A}$ with respect to $\mathcal{F}$ inf- (fin-)recognizes the same language. (That is, $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}) = \mathcal{L}_{\inf}(\mathcal{A}')$ holds for the automaton $\mathcal{A}' = \langle \Sigma, Q, \Delta', q_I, \mathcal{F}\rangle$ having

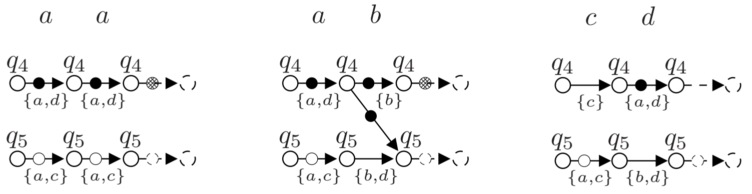


Fig. 2.4: Possible extensions of the graph in Fig. 2.3 into a run of the alternating automaton shown in Fig. 2.2 (a). (a) Extension on the input a^ω; (b) Extension on the input $(ab)^\omega$; (c) Extension on the input $(cd)^\omega$

the transition relation $\Delta' \stackrel{def}{=} \{\langle q, \Gamma, \mathcal{F} \setminus F, Q' \rangle \mid \langle q, \Gamma, F, Q' \rangle \in \Delta\}$; as a matter of fact, if $|\mathcal{F}| = 1$ holds, then fin-acceptance coincides with a notion of acceptance commonly known as *co-Büchi* acceptance.) Nevertheless, fin-acceptance provides a convenient way to identify a collection of transitions that an automaton is forbidden to take indefinitely in any infinite branch of a fin-accepting run on an input $w \in \Sigma^\omega$ belonging to the language of the automaton. Obviously, this requirement resembles the characteristic properties of models of strong temporal eventualities of LTL: recall that an infinite word does not satisfy a strong temporal eventuality if some designated LTL property remains unsatisfied in all suffixes of the word. We shall make use of this connection between the semantics of LTL and fin-acceptance in Ch. 3. Additionally, with fin-acceptance one can freely add new acceptance conditions to an automaton without modifying its transition relation to preserve the language of the automaton.

Because the different acceptance modes are reducible to each other as described above, each theorem on alternating automata working in one acceptance mode corresponds to a theorem on automata working in the opposite acceptance mode. We shall prove most of our results for only one acceptance mode and shall not deal with the opposite mode explicitly.

Example 2.3.5 Consider again the initial fragment of a run of the automaton from Ex. 2.3.1 on the input *acabacababdcd* (Fig. 2.3). This run fragment ends in a level having two nodes labeled with the states q_4 and q_5, respectively. We investigate inf- and fin-acceptance in several runs of the automaton obtained via simple infinite extensions of the input.

Concatenating the word a^ω to the input allows us to extend the graph in Fig. 2.3 into a run ending in, for example, an infinite number of identical levels shown in Fig. 2.4 (a). It is easy to see that the run formed in this way contains a finite number of infinite branches, and all of these branches end in an infinite suffix of identically labeled edges (labeled either with the transition $\langle q_4, \{a, d\}, \{\bullet\}, \{q_4\} \rangle$ or the transition $\langle q_5, \{a, c\}, \{\circ\}, \{q_5\} \rangle$). Because neither of these transitions is both a $\bullet$- and a $\circ$-transition, it follows that the conditions $\bullet$ and $\circ$ cannot both belong to the infinity set of any infinite branch, and thus the run is not inf-accepting. On the other hand, the run is not fin-accepting, either, because the final set of every infinite branch always includes one of the conditions $\bullet$ or $\circ$.

Figure 2.4 (b) shows another extension for the graph in Fig. 2.3 obtained by concatenating the word $(ab)^\omega$ with the original input. The run now contains infinitely many infinite branches: intuitively, after traversing a chain of edges that ends in an edge labeled with a self-loop starting from the state q_4, we always have the opportunity of extending this chain with an edge labeled either with another self-loop starting from q_4, or a self-loop that starts from q_5. Nevertheless, it is easy to see that all infinite branches again end in an infinite suffix of edges labeled with self-loops starting from a fixed state of the automaton. More precisely, the edge labels will eventually alternate between the transitions $\langle q_4, \{a, d\}, \{\bullet\}, \{q_4\}\rangle$ and $\langle q_4, \{b\}, \{\bullet\}, \{q_4, q_5\}\rangle$, or the transitions $\langle q_5, \{a, c\}, \{\circ\}, \{q_5\}\rangle$ and $\langle q_5, \{b, d\}, \emptyset, \{q_5\}\rangle$ in every infinite branch of the run. Similarly to the first case, no infinite branch has $\{\bullet, \circ\}$ as its infinity set, and the run is not inf-accepting. Although the final set of all infinite branches ending in a suffix labeled with self-loops starting from q_5 is empty, the run is nevertheless not fin-accepting, either, because the acceptance condition $\bullet$ will eventually repeat indefinitely in the edge labels of every infinite branch that ends in an infinite suffix of edges corresponding to self-loops starting from the state q_4.

Finally, extending the graph in Fig. 2.3 into a run by reading the input $(cd)^\omega$ can be done as shown in Fig. 2.4 (c). As above, the run is not inf-accepting; however, in this case all infinite branches of the run contain infinitely many edges labeled with transitions having no acceptance conditions as seen in the figure. Therefore the automaton fin-accepts the word $acabacababdcd(cd)^\omega$. ■

2.3.2 Properties of Runs of Alternating Automata

In this section we list several basic facts about runs of alternating automata. These facts will be used mainly as tools in the proofs of subsequent results. We begin by establishing an obvious correspondence between reachability in a run of an alternating automaton and reachability in the automaton itself; compare this result with Fig. 2.2 (a) and Fig. 2.3.

Proposition 2.3.6 *Let $G = \langle V, E, L\rangle$ be a run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ on an infinite word $w \in \Sigma^\omega$. Let $v \in V_i$ be a node in G at some level $0 \leq i < \omega$. If $v' \in V$ is a descendant of v in G, then $L(v')$ is a descendant of $L(v)$ in $\mathcal{A}$.*

Proof: Because of the partitioning of V, each descendant of v in G is an element of V_{i+j} for some $1 \leq j < \omega$. If $v' \in V_{i+1}$ is a successor of v, then v is the source state of an edge that includes v' in its target nodes, and because G is a run, the consistency of the labeling implies that $L(v')$ is a successor (hence, a descendant) of $L(v)$ in $\mathcal{A}$.

Assume that the result holds for all descendants $v' \in V_{i+j}$ of v for some $1 \leq j < \omega$, and let $v'' \in V_{i+j+1}$ be a descendant of v. Thus G contains a finite nontrivial path from v to v'', and, because E contains edges only between consecutive levels of G, there exists a descendant $v' \in V_{i+j}$ of v and an edge $e = \langle v', V'\rangle \in E$ such that $v'' \in V'$ holds. Because the labeling L is consistent, $L(v'')$ is a successor of $L(v')$ in $\mathcal{A}$, and thus $L(v'')$ is a descendant of $L(v)$ by the induction hypothesis. □

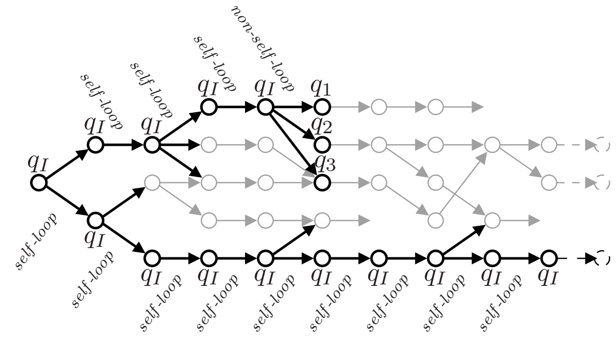


Fig. 2.5: Every run of an alternating automaton contains a chain of edges labeled with initial self-loops of the automaton such that the sequence is either infinite, or it can be extended with an edge labeled with an initial transition that is not a self-loop

Consider the construction of a run for an alternating automaton $\mathcal{A}$. Clearly, the only way to extend a finite (possibly empty) chain of edges labeled with initial self-loops of the automaton with a new edge starting from a node labeled with the initial state of the automaton in a consistent way is to label the new edge with another initial transition of the automaton. Therefore, every run of the automaton contains either a finite chain of edges labeled with initial transitions of the automaton such that all edges except the last one correspond to self-loops of the automaton, or an infinite chain of edges, all edges of which are labeled with initial self-loops of the automaton (see Fig. 2.5). Because we shall often rely on the existence of such a chain of edges in a run of an alternating automaton, we state this simple fact formally here for further reference.

Proposition 2.3.7 *Let $G = \langle V, E, L\rangle$ be a run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$. There exists an index $0 \leq i \leq \omega$ and a chain of edges $(e_j)_{0\leq j<i+1}$, $e_j \in E \cap (V_j \times 2^{V_{j+1}})$, such that $L(e_j)$ is an initial self-loop of $\mathcal{A}$ for all $0 \leq j < i$, and either $i = \omega$, or $L(e_i)$ is an initial transition of $\mathcal{A}$ that is not a self-loop.*

Proof: Because G is a run, $L(v_0) = q_I$, and v_0 has the unique outgoing edge $e = \langle v_0, V_1\rangle \in E \cap (V_0 \times 2^{V_1})$ such that $L(e)$ is an initial transition of $\mathcal{A}$. If $q_I \notin L(V_1)$, then $L(e)$ is not a self-loop, and thus $i = 0$ can be chosen as the index referred to in the proposition.

Assume that G contains a chain of edges $(e_j)_{0\leq j\leq k}$, $e_j \in E \cap (V_j \times 2^{V_{j+1}})$, for some $0 \leq k < \omega$ such that $L(e_j)$ is an initial self-loop of $\mathcal{A}$ for all $0 \leq j \leq k$. Let $e_k = \langle v, V'\rangle$ for some $v \in V_k$ and $V' \subseteq V_{k+1}$. Because $L(e_k)$ is an initial self-loop of $\mathcal{A}$, there exists a node $v' \in V'$ such that $L(v') = q_I$. Due to forward causality, v' has the unique outgoing edge $e_{k+1} = \langle v', V''\rangle \in E \cap (V_{k+1} \times 2^{V_{k+2}})$ for some $V'' \subseteq V_{k+2}$, and because L is consistent, e_{k+1} is labeled with an initial transition of $\mathcal{A}$. Clearly, e_k and e_{k+1} are consecutive. As above, if $q_I \notin L(V'')$, then $L(e_{k+1})$ is not a self-loop, and $(e_j)_{0\leq j\leq k+1}$ is a chain of edges that satisfies the criteria given in the proposition. Otherwise $(e_j)_{0\leq j\leq k+1}$ is another chain of edges labeled with initial self-loops of $\mathcal{A}$. By induction, it follows that G contains a chain of edges satisfying the required criteria. □

The following proposition proves the fact that each run of an alternating automaton $\mathcal{A}$ on an infinite word $w \in \Sigma^\omega$ is built from the runs of its subautomata on suffixes of w; compare this result again with Fig. 2.3.

Proposition 2.3.8 *Let $G = \langle V, E, L\rangle$ be a run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ on an infinite word $w \in \Sigma^\omega$. Let $v \in V_i$ be a node in G at some level $0 \leq i < \omega$ with $L(v) = q \in Q$. Define the graph $G^v = \langle V^v, E^v, L^v\rangle$, where*

- $V^v \stackrel{def}{=} \{v\} \cup \{v' \in V \mid v'$ *is a descendant of* v *in* $G\}$,
- $E^v \stackrel{def}{=} \big\{\langle v', V'\rangle \in E \bigm| \{v'\} \cup V' \subseteq V^v\big\}$, *and*
- $L^v : (V^v \cup E^v) \to (Q \cup \Delta)$ *is defined by the rule* $L^v(x) \stackrel{def}{=} L(x)$ *for all* $x \in V^v \cup E^v$.

The graph G^v is a run of the subautomaton $\mathcal{A}^q = \langle \Sigma, Q^q, \Delta^q, q, \mathcal{F}^q\rangle$ on the suffix w^i of w.

Proof: We check that G^v satisfies the properties required of a run of the subautomaton $\mathcal{A}^q$ on w^i.

(Partitioning) V^v can be partitioned into finite disjoint levels $V^v_j \stackrel{def}{=} V^v \cap V_{i+j}$ $(0 \leq j < \omega)$. Clearly, $V^v_0 = \{v\}$, and $E^v \subseteq \bigcup_{0\leq j<\omega}(V^v_j \times 2^{V^v_{j+1}})$.

(Causality) If $v' \in V^v$, then $v' = v$ or v' is a descendant of v in G, and because G is a run, there exists a unique edge $e = \langle v', V'\rangle \in E$. Clearly, every node in V' is a descendant of v. Therefore, $\{v'\} \cup V' \subseteq V^v$, and $e \in E^v$ is the unique edge starting from v' also in E^v.

If $v' \in V^v \setminus \{v\}$, then v' is either a successor of v, or a successor of a node that is itself a descendant of v in G. In either case, there exists a node $v'' \in V^v$ and an edge $e = \langle v'', V''\rangle \in E$ such that $v' \in V''$ holds. Obviously, all nodes in V'' are descendants of v. Thus $\{v''\} \cup V'' \subseteq V^v$ holds, and $e \in E^v$.

(Consistency of L^v) By the definitions of G^v and $\mathcal{A}^q$, $L^v(v) = L(v) = q$ is the initial state of $\mathcal{A}^q$. Let $v' \in V^v_j \subseteq V_{i+j}$ for some $0 \leq j < \omega$. By causality, there exists a unique edge $e = \langle v', V'\rangle \in E^v \subseteq E$ (with $V' \subseteq V_{i+j+1}$), and by consistency of L, $L(e) = \big\langle L(v'), \Gamma, F, L(V')\big\rangle \in \Delta$ holds for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w(i + j) \in \Gamma$. Because $L^v(x) = L(x)$ holds for all $x \in V^v \cup E^v$, then obviously $\big\langle L(v'), \Gamma, F, L(V')\big\rangle = \big\langle L^v(v'), \Gamma, F, L^v(V')\big\rangle = L^v(e)$ holds. It remains to check that $\big\langle L(v'), \Gamma, F, L(V')\big\rangle \in \Delta^q$. Clearly, if $v' = v$, then $L(v') = L(v) = q \in Q^q$. Otherwise v' is a descendant of v in G, and thus $L(v')$ is a descendant of $L(v) = q$ in $\mathcal{A}$ by Proposition 2.3.6. In either case, all states in $L(V')$ are descendants of $L(v)$, and thus $L\big(\{v'\} \cup V'\big) \subseteq Q^q$. Because $F \subseteq \mathcal{F} = \mathcal{F}^q$ holds by the definition of $\mathcal{A}^q$, it follows that $\big\langle L(v'), \Gamma, F, L(V')\big\rangle \in \Delta^q$ holds.

We conclude that G^v is a run of $\mathcal{A}^q$ on $\big(w(i + j)\big)_{0\leq j<\omega} = w^{[i,\omega)} = w^i$. $\square$

By focusing only on inf- or fin-accepting runs of alternating automata, Proposition 2.3.8 leads to the result that any inf- or fin-accepting run of an alternating automaton consists of inf- or fin-accepting runs of its subautomata, respectively.

Proposition 2.3.9 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton working in acceptance mode $\mu \in \{\inf, \text{fin}\}$, and let $G = \langle V, E, L \rangle$ be a μ-accepting run of $\mathcal{A}$ on $w \in \Sigma^\omega$. For all $0 \leq i < \omega$ and $v \in V_i$, the run G^v obtained from G using the construction in Proposition 2.3.8 is a μ-accepting run of $\mathcal{A}^{L(v)}$ on w^i. More generally, if $G = \langle V, E, L \rangle$ is a μ-accepting run of $\mathcal{A}$ on w, then $\mathcal{A}^{L(v)}$ μ-accepts w^i for all $0 \leq i < \omega$ and $v \in V_i$.*

Proof: By Proposition 2.3.8, G^v is a run of the subautomaton $\mathcal{A}^{L(v)}$ on w^i. If there exists an infinite path through G^v starting from the node v, then this path is a suffix of some infinite path through G that begins from the node $v_0 \in V_0$ and visits the node v. (This follows directly from backward causality and the fact that $V^v \subseteq V$ and $E^v \subseteq E$ hold.) It follows that also each infinite branch $\beta^v = (e_j^v)_{0 \leq j < \omega} \in \mathcal{B}(G^v)$ in G^v is a suffix of an infinite branch $\beta = (e_j)_{0 \leq j < \omega} \in \mathcal{B}(G)$ in G (where $e_j^v = e_{i+j}$ for all $0 \leq j < \omega$). Because G is a μ-accepting run of $\mathcal{A}$, β is μ-accepting. Thus, either $\inf(\beta) = \mathcal{F}$ ($\mu = \inf$), or $\text{fin}(\beta) = \emptyset$ ($\mu = \text{fin}$). Since β^v is an infinite suffix of β, β contains only finitely many edges not contained in β^v, and thus either $\inf(\beta^v) = \inf(\beta) = \mathcal{F} = \mathcal{F}^{L(v)}$, or $\text{fin}(\beta^v) = \text{fin}(\beta) = \emptyset$ holds. It follows that G^v is a μ-accepting run of $\mathcal{A}^{L(v)}$ on w^i. $\square$

Proposition 2.3.9 has the following immediate consequence on the non-occurrence of nodes and edges labeled with certain states or transitions of an alternating automaton in an inf- or fin-accepting run of the automaton.

Proposition 2.3.10 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton working in acceptance mode $\mu \in \{\inf, \text{fin}\}$. Let $q_\emptyset \in Q$ be a state of the automaton such that $\mathcal{L}_\mu(\mathcal{A}^{q_\emptyset}) = \emptyset$ holds, or let $t_\emptyset = \langle q, \Gamma, F, Q' \rangle \in \Delta$ ($q \in Q$, $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$) be a transition such that $\Gamma = \emptyset$, or $\bigcap_{q' \in Q'} \mathcal{L}_\mu(\mathcal{A}^{q'}) = \emptyset$ holds. For all $w \in \mathcal{L}_\mu(\mathcal{A})$, no μ-accepting run of $\mathcal{A}$ on w contains a node (edge) labeled with the state $q_\emptyset$ (the transition $t_\emptyset$, respectively).*

Proof: Let $G = \langle V, E, L \rangle$ be a μ-accepting run of $\mathcal{A}$ on some $w \in \Sigma^\omega$. If there exists a node $v \in V_i$ ($0 \leq i < \omega$) such that $L(v) = q_\emptyset$ holds, then $w^i \in \mathcal{L}_\mu(\mathcal{A}^{q_\emptyset})$ holds by Proposition 2.3.9. But then $\mathcal{L}_\mu(\mathcal{A}^{q_\emptyset})$ is not empty. It follows that G has no nodes labeled with the state $q_\emptyset$.

Let $e = \langle v, V' \rangle \in E \cap (V_i \times 2^{V_{i+1}})$ ($0 \leq i < \omega$) be an edge in G. Because the labeling L is consistent, the guard of the transition $L(e) \in \Delta$ cannot be nonempty (it contains the symbol $w(i)$), and the target nodes V' of e are labeled with the target states of $L(e)$. Because $V' \subseteq V$ holds, it follows by Proposition 2.3.9 that $\mathcal{A}^{q'}$ μ-recognizes w^{i+1} for all $q' \in L(V')$, i.e., $w^{i+1} \in \bigcap_{q' \in L(V')} \mathcal{L}_\mu(\mathcal{A}^{q'})$ holds. But then $\bigcap_{q' \in L(V')} \mathcal{L}_\mu(\mathcal{A}^{q'}) \neq \emptyset$. It is easy to see that $t_\emptyset$ cannot be the label of e. $\square$

Consequently, it is safe to remove a state (transition) from an alternating automaton if the subautomaton rooted at the state inf- or fin-recognizes the empty language (respectively, if the intersection of the languages recognized by the subautomata rooted at the transition's target states is empty).

Corollary 2.3.11 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton working in acceptance mode $\mu \in \{\inf, \text{fin}\}$, let $Q_\emptyset \subseteq Q$ be a set of states* *such that $\mathcal{L}_\mu(\mathcal{A}^q) = \emptyset$ holds for all $q \in Q$, and let $\Delta_\emptyset \subseteq \Delta$ be a set of transitions such that $\Gamma = \emptyset$ or $\bigcap_{q' \in Q'} \mathcal{L}_\mu(\mathcal{A}^{q'}) = \emptyset$ holds for all $\langle q, \Gamma, F, Q' \rangle \in \Delta_\emptyset$ ($q \in Q$, $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$). Let $\mathcal{A}' = \langle \Sigma, Q', \Delta', q_I, \mathcal{F} \rangle$ be the alternating automaton obtained from $\mathcal{A}$ by defining $Q' \stackrel{def}{=} Q \setminus \big(Q_\emptyset \setminus \{q_I\}\big)$ and $\Delta' \stackrel{def}{=} \big\{\langle q, \Gamma, F, Q'' \rangle \in \Delta \setminus \Delta_\emptyset \,\big|\, \{q\} \cup Q'' \subseteq Q' \big\}$. The automata $\mathcal{A}$ and $\mathcal{A}'$ are μ-equivalent.*

Proof: Clearly, $Q' \subseteq Q$ and $\Delta' \subseteq \Delta$ holds. Because $\mathcal{A}$ and $\mathcal{A}'$ share their initial state and their acceptance conditions, it is easy to see that every μ-accepting run of $\mathcal{A}'$ on some $w \in \Sigma^\omega$ is also a μ-accepting run of $\mathcal{A}$ on w. The converse result follows because no μ-accepting run of $\mathcal{A}$ on any $w \in \Sigma^\omega$ contains nodes or edges labeled with states from $Q_\emptyset$ or $\Delta_\emptyset$, respectively (Proposition 2.3.10). We conclude that $\mathcal{A}$ and $\mathcal{A}'$ are μ-equivalent. □

Proposition 2.3.9 implies also that the language inf- or fin-recognized by an alternating automaton depends only on the structure of the subautomaton rooted at the initial state of the automaton. Consequently, given an alternating automaton, we can always remove all its non-initial states that are not reachable from its initial state (and the transitions having such states as their source state or in their target states) without changing the language of the automaton.

Proposition 2.3.12 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton working in acceptance mode $\mu \in \{\text{inf}, \text{fin}\}$. For all $w \in \Sigma^\omega$, $\mathcal{A}$ μ-accepts w iff $\mathcal{A}^{q_I}$ μ-accepts w.*

Proof: By the definition of $\mathcal{A}^{q_I}$, the set of states (transitions) of $\mathcal{A}^{q_I}$ forms a subset of the states (transitions) of $\mathcal{A}$, and because $\mathcal{A}$ and $\mathcal{A}^{q_I}$ share their initial state and the set of acceptance conditions, every μ-accepting run of $\mathcal{A}^{q_I}$ on some $w \in \Sigma^\omega$ is also a μ-accepting run of $\mathcal{A}$ on w.

Conversely, if $G = \langle V, E, L \rangle$ (with $V_0 = \{v_0\}$) is a μ-accepting run of $\mathcal{A}$ on $w \in \Sigma^\omega$, then it follows immediately by Proposition 2.3.9 that $G^{v_0} = G$ is a μ-accepting run of $\mathcal{A}^{L(v_0)} = \mathcal{A}^{q_I}$ on $w^0 = w$. □

Example 2.3.13 Consider again the alternating automaton depicted in Fig. 2.2 (a) (p. 21). If we choose q_4 instead of q_1 as the initial state of this automaton, then, by Proposition 2.3.12, we know that the language of the automaton is completely determined by the subautomaton $\mathcal{A}^{q_4}$ shown in Fig. 2.2 (b), and thus the modified automaton and $\mathcal{A}^{q_4}$ (obtained from it by removing the states q_1, q_2 and q_3) recognize the same language. ■

2.3.3 Semi-Runs

In this section we define a class of graph structures which will be convenient for proving many results on alternating automata via transformation of runs of the automata. Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton, and let $w \in \Sigma^\omega$. We call a directed labeled acyclic graph $G = \langle V, E, L \rangle$ a *semi-run of $\mathcal{A}$ on w* iff V, E and L satisfy the partitioning, backward causality and the consistency properties defined for runs of $\mathcal{A}$ (p. 23) together with a relaxed forward causality condition defined as

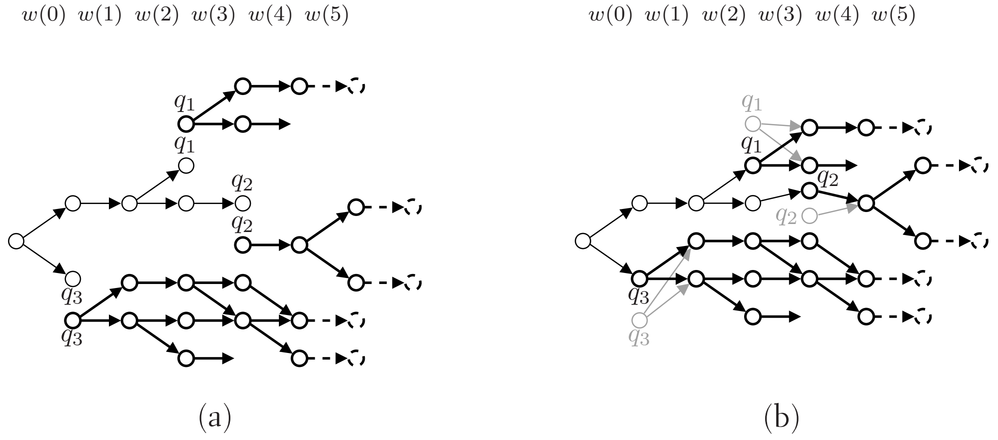


Fig. 2.6: Extending a semi-run of an automaton on an input w into a run of the automaton using runs of the automaton's subautomata. (a) A semi-run of an automaton on w together with runs of its subautomata on suffixes of w; (b) The run (black nodes and edges) obtained by joining the runs of the subautomata with the semi-run of the automaton)

For all $v \in V$, there exists at most one edge $\langle v, V' \rangle \in E$. [Forward semi-causality]

Thus, all nodes of a semi-run do not need to have any outgoing edges (but each edge starting from a node is still unique). The concepts of an infinite branch and acceptance extend to semi-runs in an obvious way: a semi-run G is called an *inf- or fin-accepting semi-run* iff each infinite branch through G is inf- or fin-accepting, respectively.

An inf- (fin-)accepting semi-run of $\mathcal{A}$ on w can be extended into an inf- (fin-)accepting run of $\mathcal{A}$ on w provided that it is possible to "attach" an inf- (fin-)accepting run of a subautomaton of $\mathcal{A}$ on a suffix of w to each node of the semi-run with no outgoing edges. This fact is formalized in the following proposition; see also Fig. 2.6.

Proposition 2.3.14 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton working in acceptance mode $\mu \in \{\text{inf}, \text{fin}\}$, and let $G = \langle V, E, L \rangle$ be a μ-accepting semi-run of $\mathcal{A}$ on $w \in \Sigma^\omega$. Let $\widehat{V}_i \stackrel{\text{def}}{=} \{v \in V_i \,|\, E \cap (\{v\} \times 2^{V_{i+1}}) = \emptyset\}$ denote the set of nodes at level $0 \leq i < \omega$ of G with no outgoing edges, and assume that $\mathcal{A}^{L(\hat{v})}$ has a μ-accepting run on w^i for all $0 \leq i < \omega$ and $\hat{v} \in \widehat{V}_i$. The automaton $\mathcal{A}$ μ-accepts w.*

Proof: For all $0 \leq i < \omega$ and all $\hat{v} \in \widehat{V}_i$, let $G^{\hat{v}} = \langle V^{\hat{v}}, E^{\hat{v}}, L^{\hat{v}} \rangle$ (with $V_0^{\hat{v}} = \{v_0^{\hat{v}}\}$) be a μ-accepting run of $\mathcal{A}^{L(\hat{v})}$ on w^i. Without loss of generality, we may assume that $V^{\hat{v}} \cap V^{\hat{v}'} = \emptyset$ holds for all pairs of nodes $\hat{v}, \hat{v}' \in \bigcup_{0 \leq i < \omega} \widehat{V}_i$ ($\hat{v} \neq \hat{v}'$) and $V^{\hat{v}} \cap V = \emptyset$ holds for all $\hat{v} \in \bigcup_{0 \leq i < \omega} \widehat{V}_i$. Define the graph $G' = \langle V', E', L' \rangle$, where

- $V' \stackrel{\text{def}}{=} V \cup \bigcup_{0 \leq i < \omega} \bigcup_{\hat{v} \in \widehat{V}_i} (V^{\hat{v}} \setminus \{v_0^{\hat{v}}\})$,
- $E' \stackrel{\text{def}}{=} E \cup \bigcup_{0 \leq i < \omega} \bigcup_{\hat{v} \in \widehat{V}_i} \big((E^{\hat{v}} \setminus \{\langle v_0^{\hat{v}}, V_1^{\hat{v}} \rangle\}) \cup \{\langle \hat{v}, V_1^{\hat{v}} \rangle\} \big)$, and

- the labeling function L' is given by

$$L'(v) \stackrel{\mathit{def}}{=} \begin{cases} L(v) & \text{if } v \in V \\ L^{\hat{v}}(v) & \text{if } v \in V^{\hat{v}} \setminus \{v_0^{\hat{v}}\} \text{ for some } \hat{v} \in \bigcup_{0 \le i < \omega} \widehat{V}_i \end{cases}$$

$$L'(e) \stackrel{\mathit{def}}{=} \begin{cases} L(e) & \text{if } e \in E \\ L^{\hat{v}}(e) & \text{if } e \in E^{\hat{v}} \setminus \big\{\langle v_0^{\hat{v}}, V_1^{\hat{v}} \rangle\big\}, \hat{v} \in \bigcup_{0 \le i < \omega} \widehat{V}_i \\ L^{\hat{v}}\big(\langle v_0^{\hat{v}}, V_1^{\hat{v}} \rangle\big) & \text{if } e = \langle \hat{v}, V_1^{\hat{v}} \rangle \text{ for some } \hat{v} \in \bigcup_{0 \le i < \omega} \widehat{V}_i \end{cases}$$

We show that G' is a μ-accepting run of $\mathcal{A}$ on w by checking that G' satisfies all properties required of an accepting run.

(Partitioning) V' can be partitioned into finite disjoint levels by defining $V'_i \stackrel{\mathit{def}}{=} V_i \cup \bigcup_{0 \le j < i} \bigcup_{v \in \widehat{V}_j} V_{i-j}^{v}$; then also $E' \subseteq \bigcup_{i=0}^{\omega} (V'_i \times 2^{V'_{i+1}})$ holds. In addition, because G is a semi-run of $\mathcal{A}$, $V'_0 = V_0$ is a singleton.

(Forward causality) Let $v \in V'$. If $v \in V$, and v has an outgoing edge in G, then forward causality follows because $E \subseteq E'$ holds. Otherwise, if $v \in V$ has no outgoing edges in G, then $v \in \widehat{V}_i$ for some $0 \le i < \omega$, and by the definition of G', $\langle v, V_1^{\hat{v}} \rangle$ is the unique edge starting from v in this case. In the remaining case, $v \in V^{\hat{v}} \setminus \{v_0^{\hat{v}}\}$ holds for some $0 \le i < \omega$ and $\hat{v} \in \widehat{V}_i$. Because $G^{\hat{v}}$ is a run of $\mathcal{A}^{L(\hat{v})}$ on w^i, there exists a unique edge $e = \langle v, V' \rangle \in E^{\hat{v}}$, and because $v \neq v_0^{\hat{v}}$, $e \in E'$ holds by the definition of G'.

(Backward causality) Let $v' \in V' \setminus V'_0$. If $v' \in V$, then, because G is a semi-run, there exists (in G) a node $v \in V$ and an edge $e \in E$ that starts from v and includes v' in its target nodes. Because $V \subseteq V'$ and $E \subseteq E'$, the same still holds in G'. If $v' \in V^{\hat{v}} \setminus \{v_0^{\hat{v}}\}$ for some $\hat{v} \in \bigcup_{0 \le i < \omega} \widehat{V}_i$, then $v' \in V_j^{\hat{v}}$ for some $1 \le j < \omega$. Because $G^{\hat{v}}$ is a run, there exists (in $G^{\hat{v}}$) a node $v \in V_{j-1}^{\hat{v}}$ and an edge $e \in E^{\hat{v}}$ that starts from v and includes v' in its target nodes. If $j > 1$, then $v \in V^{\hat{v}} \setminus \{v_0^{\hat{v}}\} \subseteq V'$ and $e \in E^{\hat{v}} \setminus \big\{\langle v_0^{\hat{v}}, V_1^{\hat{v}} \rangle\big\} \subseteq E'$, and backward causality follows. If $j = 1$, then v' is a successor of a node $v \in \bigcup_{0 \le i < \omega} \widehat{V}_i$ by the definition of G', and G' has the backward causality property also in this case.

(Consistency of L') Because $V'_0 = V_0 = \{v_0\}$ and L is consistent, $L'(v_0) = L(v_0) = q_I$.

Let $e \in E'$. Clearly, $e \in V'_i \times 2^{V'_{i+1}}$ for some $0 \le i < \omega$. If $e \in E$ holds, then $e = \langle v, V' \rangle \in V_i \times 2^{V_{i+1}}$ holds by the definition of G'. Because L is consistent, it follows from the definition of G' that $L'(e) = L(e) = \big\langle L(v), \Gamma, F, L(V') \big\rangle \in \Delta$ holds in G for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w(i) \in \Gamma$. The labeling L' is now consistent because $\big\langle L(v), \Gamma, F, L(V') \big\rangle = \big\langle L'(v), \Gamma, F, L'(V') \big\rangle$.

If $e = \langle \hat{v}, V_1^{\hat{v}} \rangle$ holds for some $\hat{v} \in \widehat{V}_i$, then, by the definition of G' and the consistency of $L^{\hat{v}}$, $L'(e) = L^{\hat{v}}\big(\langle v_0^{\hat{v}}, V_1^{\hat{v}} \rangle\big) = \big\langle L^{\hat{v}}(v_0^{\hat{v}}), \Gamma, F, L^{\hat{v}}(V_1^{\hat{v}}) \big\rangle \in \Delta$ holds for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w^i(0) = w(i) \in \Gamma$. Then also the labeling L' is consistent, because $\big\langle L^{\hat{v}}(v_0^{\hat{v}}), \Gamma, F, L^{\hat{v}}(V_1^{\hat{v}}) \big\rangle = \big\langle L(\hat{v}), \Gamma, F, L^{\hat{v}}(V_1^{\hat{v}}) \big\rangle = \big\langle L'(\hat{v}), \Gamma, F, L'(V_1^{\hat{v}}) \big\rangle$.

Finally, if $e = \langle v, V' \rangle \in E^{\hat{v}} \setminus \big\{\langle v_0^{\hat{v}}, V_1^{\hat{v}} \rangle\big\}$ holds for some $\hat{v} \in \widehat{V}_i$ ($0 \le i < \omega$), then $e \in V_j^{\hat{v}} \times 2^{V_{j+1}^{\hat{v}}} \subseteq V'_{i+j} \times 2^{V'_{i+j+1}}$ holds for some $1 \le j < \omega$ (and $\{v\} \cup V' \subseteq V^{\hat{v}} \setminus \{v_0^{\hat{v}}\}$). Because $G^{\hat{v}}$ is a run of $\mathcal{A}^{L(\hat{v})}$ on w^i, the labeling $L^{\hat{v}}$ is consistent, and thus $L'(e) = L^{\hat{v}}(e) = \big\langle L^{\hat{v}}(v), \Gamma, F, L^{\hat{v}}(V') \big\rangle \in \Delta$ holds for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w^i(j) = w(i+j) \in \Gamma$. Because $\langle L^{\hat{v}}(v), \Gamma, F, L^{\hat{v}}(V') \rangle = \langle L'(v), \Gamma, F, L'(V') \rangle$, L' is consistent also in this case.

(Acceptance) Clearly, G' is μ-accepting if all branches of G' are finite. Otherwise let $\beta \in \mathcal{B}(G')$ be an infinite branch in G'. If β is contained in G, then β is μ-accepting by assumption. Otherwise the branch consists of a finite (possibly empty) chain of edges in G followed by an edge of the form $e = \langle \hat{v}, V_1^{\hat{v}} \rangle$ for some $\hat{v} \in \bigcup_{0 \leq i < \omega} \widehat{V}_i$ and $V_1^{\hat{v}} \subseteq V^{\hat{v}}$, which is then followed by an infinite chain of edges through $G^{\hat{v}}$. Clearly, this chain is a suffix of an infinite branch $\beta^{\hat{v}}$ in $G^{\hat{v}}$. Since the number of edges in β preceding this suffix is finite, it follows that $\inf(\beta) = \inf(\beta^{\hat{v}})$ and $\mathrm{fin}(\beta) = \mathrm{fin}(\beta^{\hat{v}})$, and since either $\inf(\beta^{\hat{v}}) = \mathcal{F}$ or $\mathrm{fin}(\beta^{\hat{v}}) = \emptyset$ (depending on the acceptance mode μ), it follows that also $\inf(\beta) = \mathcal{F}$ or $\mathrm{fin}(\beta) = \emptyset$ holds. We conclude that G' is a μ-accepting run of $\mathcal{A}$ on w. □

Proposition 2.3.14 provides a simple method for constructing accepting runs for alternating automata by first finding an accepting semi-run for the automaton, and then extending it with accepting runs for the automaton's subautomata. We shall make extensive use of this result in later chapters. As a first example of an application of the result, we establish a simple correspondence between a single step of operation of an alternating automaton and inf- (fin-)acceptance. More precisely, an alternating automaton inf- (fin-)accepts its input only if all copies of the automaton spawned by the first transition taken by the automaton inf- (fin-)accept the input that remains after the first input symbol; conversely, the automaton can always be made to accept its input if it has such an initial transition whose guard contains the first symbol of the input.

Proposition 2.3.15 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton working in acceptance mode $\mu \in \{\inf, \mathrm{fin}\}$, and let $w \in \Sigma^\omega$. The automaton $\mathcal{A}$ μ-accepts w iff there exists a transition $\langle q_I, \Gamma, F, Q' \rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$ such that $w(0) \in \Gamma$ holds, and for all $q \in Q'$, the subautomaton $\mathcal{A}^q$ μ-accepts w^1.*

Proof: (Only if) Assume that $\mathcal{A}$ μ-accepts w. Then $\mathcal{A}$ has a μ-accepting run $G = \langle V, E, L \rangle$ on w, and G contains a state $v_0 \in V_0$ labeled with the initial state of $\mathcal{A}$ and an edge $e = \langle v_0, V_1 \rangle \in E$ labeled with a transition $t = \langle q_I, \Gamma, F, Q' \rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w(0) \in \Gamma$ and $Q' = L(V_1)$ hold. By Proposition 2.3.8, we can extract from G a run of the subautomaton $\mathcal{A}^q$ on w^1 for all $q \in Q'$, and because G is μ-accepting, each of these runs is μ-accepting by Proposition 2.3.9.

(If) Assume that there exists a transition $t = \langle q_I, \Gamma, F, Q' \rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' = \{q_1, \ldots, q_n\} \subseteq Q$ $(0 \leq n < \omega)$ such that $w(0) \in \Gamma$ holds, and the subautomaton $\mathcal{A}^{q_i}$ has a μ-accepting run on w^1 for all $1 \leq i \leq n$. Define the graph $G = \langle V, E, L \rangle$, where

- $V \stackrel{\mathit{def}}{=} \{v_I, v_1, \ldots, v_n\}$, where $v_I \neq v_i$ for all $1 \leq i \leq n$, and $v_i \neq v_j$ for all $1 \leq i, j \leq n$, $i \neq j$,

- $E \stackrel{def}{=} \{\langle v_I, \{v_1, \ldots, v_n\}\rangle\}$, and
- $L(v_I) \stackrel{def}{=} q_I$, $L(v_i) \stackrel{def}{=} q_i$ for $1 \leq i \leq n$, and $L(\langle v_I, \{v_1, \ldots, v_n\}\rangle) \stackrel{def}{=} t$.

It is easy to see from the definitions that G is a semi-run of $\mathcal{A}$ on w: obviously, V is partitioned into finite disjoint levels $V_0 = \{v_I\}$, $V_1 = \{v_1, \ldots, v_n\}$, $V_i = \emptyset$ for all $2 \leq i < \omega$ (and $E \subseteq V_0 \times 2^{V_1}$), the edge starting from v_I is unique (and it is the only edge in E), each node $v \in V \setminus \{v_I\}$ is a successor of v_I, and the labeling L is consistent. Since G has no infinite branches, G is trivially μ-accepting. Because $L(v) \in \{q_1, \ldots, q_n\}$ holds for all nodes $v \in V$ with no outgoing edges (i.e., for all $v \in V_1$) and $\mathcal{A}^{q_i}$ has a μ-accepting run on w^1 for all $1 \leq i \leq n$ by the assumption, we can apply Proposition 2.3.14 to extend the semi-run G into a μ-accepting run of $\mathcal{A}$ on w. □

2.3.4 Self-loop Alternating Automata

In this work we concentrate on a restricted class of alternating automata known to be closely related to linear time temporal logic in that every language definable as the set of models of an LTL formula is also a language recognizable by an alternating automaton in this subclass and vice versa [Rohde 1997; Löding and Thomas 2000]. Since automata in general possess intuitively appealing "operational" characteristics, translating linear time temporal logic into finite automata provides a first step towards effective procedures, for example, for checking the satisfiability of LTL formulas. By concentrating on a subclass of automata that is equally expressive to LTL, the characteristic properties of these automata allow making the procedures simpler and more efficient. Because our basic definitions of automata and acceptance generalize traditional definitions by allowing alternating automata to have multiple acceptance conditions associated with their transitions, we shall rephrase several basic results on this subclass of alternating automata in this and the following chapter using the generalized definitions to provide explicit details of various automata constructions. These details are needed, for example, for transforming the formal constructions into an actual implementation.

Definition and Relation to Weak Alternating Automata

Formally, we call an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ a *self-loop alternating automaton* iff all its simple cycles (i.e., cycles that do not visit any state except their first state twice) are self-loops.

Self-loop alternating automata share their structural properties with a subclass of alternating automata referred to as the class of *very weak* [Isli 1994, 1996; Rohde 1997; Gastin and Oddoux 2001], *linear* [Löding and Thomas 2000], *linear weak* [Merz and Sezgin 2003; Hammer et al. 2005] or *one-weak* [Ben-David et al. 2005] alternating automata in the literature. The usual terminology stems from the identification of the subclass with a special case of the more general *weak* alternating automata introduced by Muller et al. [1986, 1992]; however, we prefer the above direct definition to separate the intuitive structural characterization of the automata in the subclass from any particular (usually state-based) notion of acceptance that is implied by the standard definition of weakness.

Weak alternating automata [Muller et al. 1986, 1992] have their state sets partitioned into subsets arranged into a partially ordered hierarchy in which no state in any subset of the hierarchy has a successor that belongs to a set that is strictly higher in the hierarchy. In very weak automata, each of these subsets consists of a single state of the automaton. The connection between this structural property of very weak alternating automata and self-loop automata is given in the following proposition.

Proposition 2.3.16 *An alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ is a self-loop alternating automaton iff there exists a mapping $\rho : Q \to \mathbb{N}$ such that for all transitions $\langle q, \Gamma, F, Q' \rangle \in \Delta$ ($q \in Q$, $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$), $\rho(q') < \rho(q)$ holds for all $q' \in Q' \setminus \{q\}$.*

Proof: (Only if) Assume that all simple cycles in the automaton $\mathcal{A}$ are self-loops. Let $Q_0 \subseteq Q$ denote the set of states such that for all $q \in Q_0$, q either has no successors, or the only successor of q is q itself. We claim that for all $q \in Q$, either $q \in Q_0$ holds, or q has a descendant $q' \in Q_0$. If this were not the case, there would exist a state $q \in Q \setminus Q_0$ with no descendants in Q_0. Therefore, the automaton would contain an infinite path $(q_i)_{0 \le i < \omega}$ with $q_0 = q$ and $q_i \neq q_{i+1}$ for all $0 \le i < \omega$. Since Q is finite, there would now exist two indices $0 \le n < m < \omega$ such that $q_{n+1} \neq q_n$ and $q_m = q_n$ hold. However, the automaton would then contain a simple cycle (from q_n to itself) that is not a self-loop, contrary to the assumption.

The above result shows that the mapping $\rho : Q \to \mathbb{N}$,

$$\rho(q) \stackrel{def}{=} \max \big\{ |x| \bigm| x \text{ is a simple path from } q \text{ to a state } q' \in Q_0 \big\}$$

is well-defined on Q.

To show that this function satisfies the criterion given in the proposition, suppose that $\mathcal{A}$ contains a state $q \in Q$ and a transition $\langle q, \Gamma, F, Q' \rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$ such that $\rho(q') \ge \rho(q)$ holds for some $q' \in Q' \setminus \{q\}$. Then there exists a simple path of length $\rho(q')$ from q' to a state $q'' \in Q_0$ in the automaton. However, because $q' \neq q$ is a successor of q, there exists (due to the fact that all simple cycles of $\mathcal{A}$ are self-loops) a simple path of length $\rho(q') + 1$ from q to q''. But then $\rho(q)$ cannot be the maximal length of a simple path from q to a state in Q_0. Thus, $\rho(q') < \rho(q)$ holds.

(If) Assume that there exists a mapping $\rho : Q \to \mathbb{N}$ satisfying the given criterion. Let $x = (q_i)_{0 \le i \le n}$ ($1 \le n < \omega$) be a simple cycle in $\mathcal{A}$. Suppose that the cycle is not a self-loop, i.e., $n > 1$ holds. Since q_{i+1} is a successor of q_i for all $0 \le i < n$ and all states $\{q_0, \ldots, q_{n-1}\}$ are distinct (and $q_{n-1} \neq q_n = q_0$), it follows that

$$\rho(q_0) > \rho(q_1) > \cdots > \rho(q_{n-1}) > \rho(q_n) = \rho(q_0),$$

which is clearly a contradiction. Therefore $n = 1$, and x is a self-loop. It follows that $\mathcal{A}$ is a self-loop alternating automaton. □

Example 2.3.17 Figure 2.7 depicts a self-loop alternating automaton working over the alphabet $\{a, b, c\}$. The function ρ (defined in the proof of Proposition 2.3.16) divides the state set of the automaton into four partitions

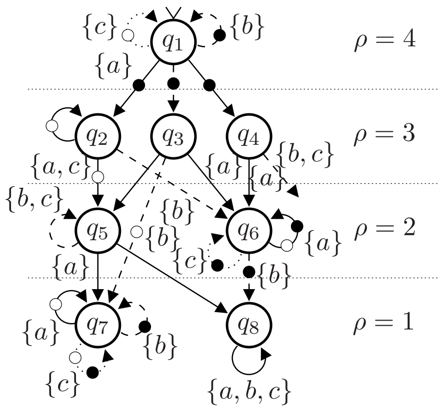


Fig. 2.7: A self-loop alternating automaton

as shown in the figure. The structure defined by the states and transitions of a self-loop alternating automaton can be considered to have been built from a directed acyclic graph by adding to it edges including their own source state in their target states. ■

Convergence of Infinite Run Branches

Because every loop of a self-loop alternating automaton visits a single state, the labels of the source states of the edges in an infinite branch of a run of the automaton will eventually converge to a fixed state of the automaton. In this case we simply say that *the branch converges to a fixed state of the automaton.*

Proposition 2.3.18 *Let $G = \langle V, E, L\rangle$ be a run of a self-loop alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$. For each infinite branch $(e_i)_{0\leq i<\omega} = \big(\langle v_i, V_i'\rangle\big)_{0\leq i<\omega} \in \mathcal{B}(G)$, there exists an index $0 \leq j < \omega$ and a state $q \in Q$ such that for all $j \leq k < \omega$, $L(v_k) = q$ holds, and $L(e_k)$ is a self-loop transition of $\mathcal{A}$ with source state q.*

<u>Proof:</u> Let $\rho : Q \to \mathbb{N}$ be a mapping satisfying the condition given in Proposition 2.3.16. By the definition of a run, $L(e_i)$ is a transition of $\mathcal{A}$ having source state $L(v_i)$ and including $L(v_{i+1})$ in its target states for all i. It follows that $\big(\rho\big(L(v_i)\big)\big)_{0\leq i<\omega}$ is a nonincreasing infinite sequence of nonnegative integers, and thus there exists an index $0 \leq j < \omega$ such that $\rho\big(L(v_k)\big) = \rho\big(L(v_j)\big)$ holds for all $j \leq k < \omega$. But then also $L(v_k) = L(v_j)$ holds for all $j \leq k < \omega$, since $L(v_{i+1})$ is a successor of $L(v_i)$ in $\mathcal{A}$ for all $0 \leq i < \omega$, and $\rho(q')$ is strictly less than $\rho\big(L(v_j)\big)$ for all successors q' of $L(v_j)$ other than $L(v_j)$ itself. Thus we may choose $q = L(v_j)$, and because $L(v_{k+1}) = q$ is included in $L(e_k)$'s target states for all $j \leq k < \omega$, it follows that $L(e_k)$ is a self-loop of $\mathcal{A}$ with source state q for all $j \leq k < \omega$. □

A state $q \in Q$ of a self-loop alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ working in fin-acceptance mode is a *transient* state iff all self-loops from this state to itself share a common acceptance condition, formally, if there exists an acceptance condition $f \in \mathcal{F}$ such that $f \in F$ holds for all $\langle q, \Gamma, F, Q'\rangle \in \Delta$ with $q \in Q'$. (This holds trivially if there are no self-loops starting from the state.)

It is easy to show that every infinite branch of a fin-accepting run of a self-loop alternating automaton will converge to a nontransient state of the automaton.

Corollary 2.3.19 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton, and let $G = \langle V, E, L \rangle$ be a run of $\mathcal{A}$ on $w \in \Sigma^\omega$. If G is a fin-accepting run of $\mathcal{A}$, then each infinite branch of G converges to a nontransient state of $\mathcal{A}$.*

Proof: Let $\beta = (e_i)_{0 \leq i < \omega} = \big(\langle v_i, V'_i \rangle\big)_{0 \leq i < \omega} \in \mathcal{B}(G)$ be an infinite branch in G. By Proposition 2.3.18, there exists a state $q \in Q$ and an index $0 \leq j < \omega$ such that for all $j \leq k < \omega$, $L(v_k) = q$, and $L(e_k)$ is a self-loop of $\mathcal{A}$ with source state q. Because G is fin-accepting, $\mathrm{fin}(\beta) = \emptyset$, and thus, for all $f \in \mathcal{F}$ and $j \leq k < \omega$, there exists a $k \leq k' < \omega$ such that the self-loop $L(e_{k'}) \in \Delta$ is not an f-transition of $\mathcal{A}$. Because the same holds for all acceptance conditions in $\mathcal{F}$, it follows that q is a nontransient state of $\mathcal{A}$. □

Another corollary of Proposition 2.3.18 is that no acceptance condition associated with a non-self-loop transition of a self-loop alternating automaton affects the language recognized by the automaton. Thus, we can always remove all acceptance conditions from the transitions of the automaton which are not self-loops.

Corollary 2.3.20 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton. Define the self-loop alternating automaton $\mathcal{A}' = \langle \Sigma, Q, \Delta', q_I, \mathcal{F} \rangle$, where Δ' is obtained from Δ by making the set of acceptance conditions of all non-self-loops of Δ empty (formally, $\Delta' \stackrel{\text{def}}{=} \big\{ \langle q, \Gamma, F, Q' \rangle \in \Delta \,\big|\, q \in Q' \big\} \cup \big\{ \langle q, \Gamma, \emptyset, Q' \rangle \,\big|\, \langle q, \Gamma, F, Q' \rangle \in \Delta$ for some $F \subseteq \mathcal{F}$, and $q \notin Q' \big\}$). The automata $\mathcal{A}$ and $\mathcal{A}'$ are fin-equivalent.*

Proof: Let $\beta = (e_i)_{0 \leq i < \omega} \in \mathcal{B}(G)$ be an infinite branch through a run G of either of the automata. By Proposition 2.3.18, there exists an index $0 \leq j < \omega$ such that $L(e_i)$ is a self-loop of both automata for all $j \leq i < \omega$, and thus β contains only finitely many edges labeled with non-self-loop transitions. Therefore, none of these transitions can contribute to the acceptance conditions occurring infinitely often in the labels of the edges of β, i.e., $\mathrm{inf}(\beta) = \mathrm{inf}\,\big((e_i)_{j \leq i < \omega}\big)$ and $\mathrm{fin}(\beta) = \mathrm{fin}\big((e_i)_{j \leq i < \omega}\big)$. The result now follows since the definitions of $\mathcal{A}'$ and $\mathcal{A}$ differ only in the acceptance conditions associated with non-self-loop transitions. □

Example 2.3.21 Consider again the automaton shown in Fig. 2.7. By Corollary 2.3.20, we can remove all acceptance conditions from the non-self-loop transitions of the automaton. This simplification results in the automaton shown in Fig. 2.8. Because both the original and the simplified automaton have no self-loops starting from the states q_3 or q_4, these states are trivially transient. Also the states q_2 and q_6 are transient, because all self-loops starting from these states share a common acceptance condition. Thus, by Corollary 2.3.19, every infinite branch in a fin-accepting run of either automaton converges to one of the states q_1, q_5, q_7 or q_8. ■

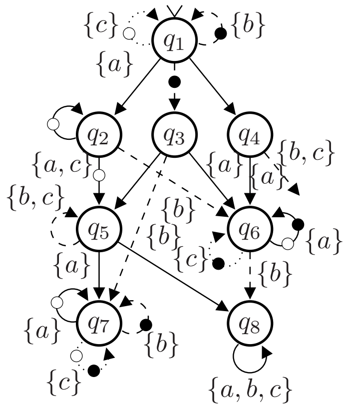


Fig. 2.8: The self-loop alternating automaton of Fig. 2.7 after removing acceptance conditions from its non-self-loop transitions

3 BASIC AUTOMATON TRANSLATION

The languages definable by linear time temporal logic formulas are known to be recognizable by nondeterministic automata on infinite words, i.e., every linear time temporal logic formula can be translated into a corresponding nondeterministic automaton that accepts its models. This connection between LTL and automata theory has stimulated active research towards finding an efficient procedure for translating LTL and its subclasses or extensions into automata. The proposed approaches can be roughly divided into (i) procedures based on combining a "local automaton" with an "eventuality automaton" [Wolper et al. 1983; Vardi and Wolper 1986; Ramakrishna et al. 1992b; Vardi and Wolper 1994], (ii) procedures that build on a tableau decision procedure for LTL ([Manna and Wolper 1982, 1984; Wolper 1985]) [Gerth et al. 1995; Couvreur 1999; Daniele et al. 1999; Somenzi and Bloem 2000; Wolper 2001; Giannakopoulou and Lerda 2002; Thirioux 2002], and (iii) translation procedures that build an automaton incrementally by combining automata built for subformulas of an LTL formula into more complex automata, using nondeterministic [Michel 1985; de Jong 1992; Schneider 2001; Fritz 2005] or alternating [Isli 1994, 1996; Vardi 1994; Rohde 1997; Manna and Sipma 2000; Gastin and Oddoux 2001; Fritz 2003; Hammer et al. 2005] automata as the target formalism. The translation procedures have been improved with techniques for the minimization of automata [Etessami and Holzmann 2000; Somenzi and Bloem 2000; Etessami et al. 2001, 2005; Etessami 2002; Fritz and Wilke 2002, 2005; Gurumurthy et al. 2002] and heuristics for reducing nondeterminism in the generated automata [Thirioux 2002; Sebastiani and Tonetta 2003]. The procedures have also been specialized for LTL safety properties [Geilen 2001; Latvala 2003].

The language of any LTL formula can be recognized by a nondeterministic automaton having exponentially many states in the length of the formula [Wolper et al. 1983; Vardi and Wolper 1994], and this upper bound is tight (see, for example, [Wolper 2001]). Muller et al. [1988] proposed using weak alternating automata as a succinct automata-theoretic formalism for working with many temporal logics, showing (using the extended temporal logic of Wolper [1981, 1983] interpreted over tree models as an example) them to be translatable into automata with only a linear number of states in the length of a formula; the special case for LTL has since been discussed in many sources [Isli 1994, 1996; Vardi 1994; Rohde 1997; Manna and Sipma 2000; Gastin and Oddoux 2001; Fritz 2003; Hammer et al. 2005]. Although all translations from LTL to alternating automata are very similar, they use slightly different strategies for dealing with negations in the input formulas. Common approaches include working directly with the *closure* of the input formula [Isli 1994, 1996; Vardi 1994] (i.e., a set of formulas obtained from the subformulas of the formula and their negations), rewriting the formula in positive normal form before translation [Manna and Sipma 2000; Gastin and Oddoux 2001; Fritz 2003], or using a complementation procedure for alternating automata [Rohde 1997].

In this chapter we describe a translation from linear time temporal logic to self-loop alternating automata working in fin-acceptance mode. Borrowing ideas from known translation procedures [Rohde 1997; Gastin and Oddoux 2001], we use a set of rules to construct a self-loop alternating automaton $\mathcal{A}_\varphi$ that recognizes the language of a given LTL formula φ (Sect. 3.1). The translation proceeds in a bottom-up manner by joining automata built recursively for subformulas of (the positive normal form of) φ into increasingly complex automata. This intuitive "modular" approach to building automata was proposed already in some studies on translating LTL formulas directly into nondeterministic automata [Michel 1985; de Jong 1992] (and applied also by Schneider [2001] and Fritz [2005]); however, these constructions are made complicated by the intricacies of working with nondeterministic infinite word automata in general, such as the difficulty of their complementation. These complexities do not arise when using alternating automata as the target formalism.

We show that the worst-case number of states in the automaton built using our translation rules meets the best upper bound known for similar translations presented in the literature (Sect. 3.2) and show the correctness of the translation (Sect. 3.3). Although formally only a matter of preference, using fin-acceptance instead of inf-acceptance (a direct generalization of the idea of using co-Büchi acceptance as suggested by Gastin and Oddoux [2001]) gives a simple explanation for the introduction of new acceptance conditions during the translation. Finally, we shall review the connection between the expressiveness of LTL and self-loop alternating automata [Rohde 1997; Löding and Thomas 2000] by discussing a *reverse* translation from self-loop alternating automata to LTL and analyzing its complexity (Sect. 3.4).

3.1 TRANSLATION RULES

In this section we introduce rules for translating LTL formulas into self-loop alternating automata. We first review the notation that is customarily used to simplify the representation of transition guards of automata working on inputs over the fixed alphabet 2^{AP}. With this alphabet, the transition guards will be elements of the set $2^{2^{AP}}$, i.e., families of sets of atomic propositions. Since there is a simple correspondence between these families and Boolean formulas, it is convenient to express the guards with these formulas. More specifically, for any family $\Gamma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \in 2^{2^{AP}}$ $(0 \leq n < \omega)$, where $\sigma \subseteq AP$ for all $1 \leq i \leq n$, there exists a *characteristic Boolean formula* θ_Γ, for example, $\theta_\Gamma \overset{def}{=} \bigvee_{i=1}^{n} \big((\bigwedge_{p\in\sigma_i} p) \wedge (\bigwedge_{p\in AP\setminus\sigma_i} \neg p)\big)$, such that, given a subset $\sigma \subseteq AP$, $\sigma \models \theta_\Gamma$ holds iff $\sigma \in \Gamma$; conversely, each Boolean formula θ is characteristic for the family of its models $\Gamma_\theta \overset{def}{=} \{\sigma \subseteq AP \mid \sigma \models \theta\} \in 2^{2^{AP}}$. Therefore, when considering the runs of an alternating automaton, the fact that $\sigma \in \Gamma$ holds for some $\sigma \subseteq AP$ and some guard $\Gamma \in 2^{2^{AP}}$ of some transition is equivalent to the condition that $\sigma \models \theta$ holds for a characteristic Boolean formula θ of Γ. This notation will be used in further discussion whenever dealing with automata having the fixed alphabet 2^{AP}.

Let $\varphi \in LTL(AP)$ be an LTL formula. By the discussion in Sect. 2.2.3, we may assume that φ is in positive normal form (by first replacing φ with $[\varphi]^{\mathrm{PNF}}$ if necessary). We construct from φ an alternating automaton $\mathcal{A}_\varphi$ by applying the following rules recursively to the subformulas of φ. See Fig. 3.1

for illustration on the application of each rule.

Atomic Formulas
Let $\varphi \in \{\top, \bot\}$ or $\varphi \in \{p, \neg p\}$ for some atomic proposition $p \in AP$. The automaton for φ is defined as $\mathcal{A}_\varphi = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$, where $Q \stackrel{def}{=} \{q_I\}$ for some new state q_I,

$$\Delta \stackrel{def}{=} \begin{cases} \big\{\langle q_I, \varphi, \emptyset, \emptyset\rangle\big\} & \text{if } \varphi \in \{\top, p, \neg p\}\ (p \in AP) \\ \emptyset & \text{otherwise} \end{cases}$$

and $\mathcal{F} \stackrel{def}{=} \emptyset$.

Next Time
Let $\varphi = \mathsf{X}\varphi_1$. Given the definition of the automaton $\mathcal{A}_{\varphi_1} = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$ for the subformula φ_1, the automaton $\mathcal{A}_\varphi = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ for φ has the components

- $Q \stackrel{def}{=} Q_1 \cup \{q_I\}$ (where q_I is a state not included in Q_1);
- $\Delta \stackrel{def}{=} \Delta_1 \cup \big\{\langle q_I, \top, \emptyset, \{q_{I1}\}\rangle\big\}$; and
- $\mathcal{F} \stackrel{def}{=} \mathcal{F}_1$.

Binary Connectives
Let $\varphi = (\varphi_1 \circ \varphi_2)$ for some binary connective $\circ \in \{\vee, \wedge, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$. Let $\mathcal{A}_{\varphi_1} = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$ and $\mathcal{A}_{\varphi_2} = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2\rangle$ be already defined for the top-level subformulas φ_1 and φ_2 of φ, respectively, such that $\mathcal{A}_{\varphi_1}^{q, \mathcal{F}_1 \cup \mathcal{F}_2} = \mathcal{A}_{\varphi_2}^{q, \mathcal{F}_1 \cup \mathcal{F}_2}$ holds for all $q \in Q_1 \cap Q_2$ (i.e., if the two automata share a state, then they share all states and transitions reachable from this state). The automaton $\mathcal{A}_\varphi = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ for the formula φ is built by defining

- $Q \stackrel{def}{=} Q_1 \cup Q_2 \cup \{q_I\}$, where q_I is a new state not included in $Q_1 \cup Q_2$;
- $\Delta \stackrel{def}{=} \Delta_1 \cup \Delta_2 \cup \Delta_\circ$; and
- $\mathcal{F} \stackrel{def}{=} \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_\circ$

where the definitions of $\Delta_\circ$ and $\mathcal{F}_\circ$ for each binary connective are given in Table 3.1.

Example 3.1.1 We illustrate the use of the translation rules by building an automaton for the LTL formula

$$\Big(\big(\mathsf{GF}p_1 \wedge \mathsf{GF}p_2\big) \vee \big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)\Big) \in LTL\big(\{p_1, \ldots, p_5\}\big).$$

Because we do not have explicit translation rules for the F and G connectives, we first rewrite the subformulas with F or G as their main connective in terms of the basic connectives via the LTL identities

$$\mathsf{F}\varphi \equiv (\top \,\mathsf{U_s}\, \varphi) \qquad \text{and} \qquad \mathsf{G}\varphi \equiv (\bot \,\mathsf{R_w}\, \varphi).$$

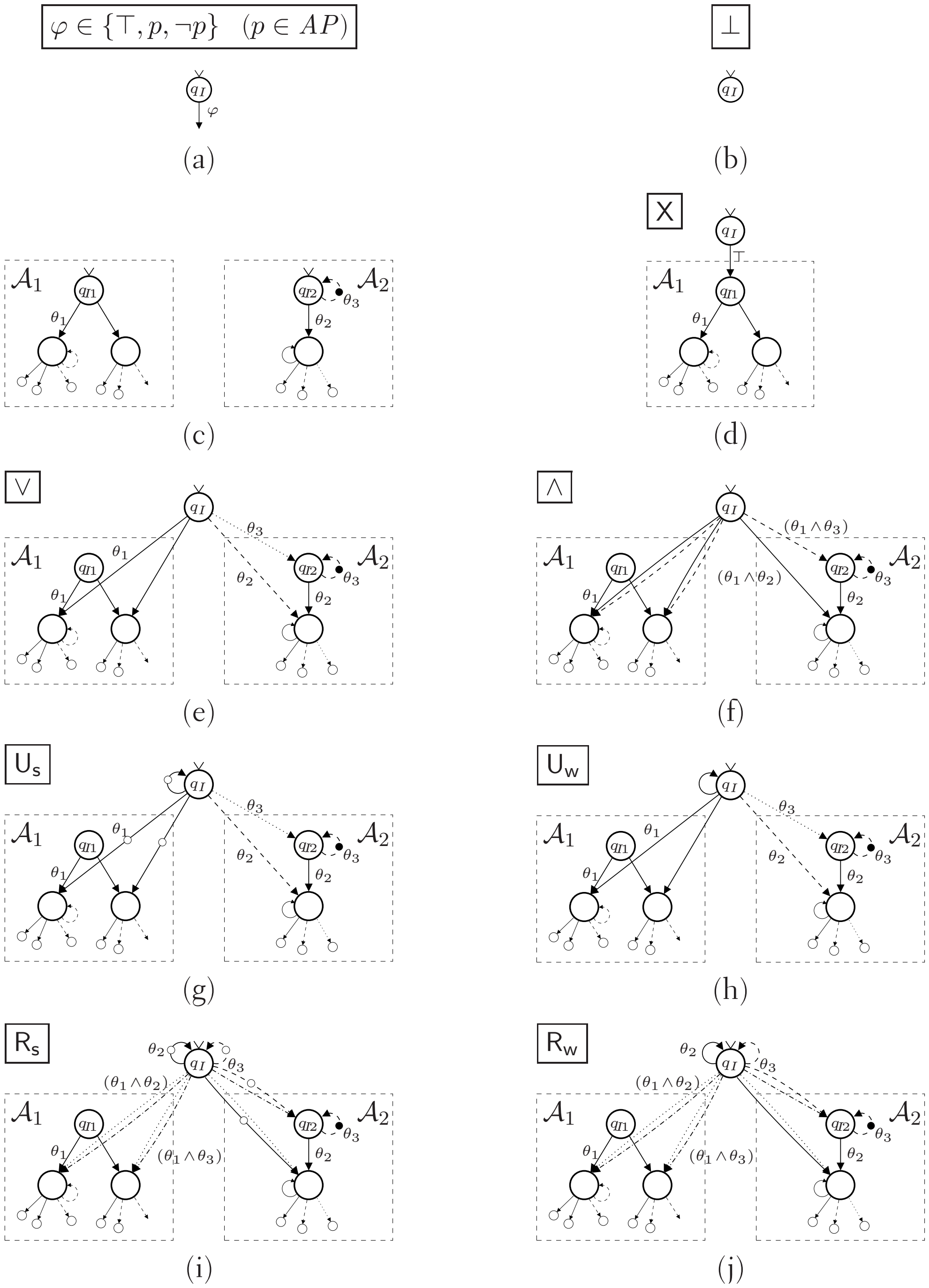


Fig. 3.1: Automata built using translation rules. (a) Automaton built from an atomic formula $\varphi \in \{\top, p, \neg p\}$ ($p \in AP$); (b) Automaton built for the atomic formula $\perp$; (c) Two component automata $\mathcal{A}_1$ and $\mathcal{A}_2$; (d) Automaton built from $\mathcal{A}_1$ with the Next Time rule; (e)–(j) Automata built from $\mathcal{A}_1$ and $\mathcal{A}_2$ using the translation rules given for the $\vee$, $\wedge$, $\mathsf{U_s}$, $\mathsf{U_w}$, $\mathsf{R_s}$ and $\mathsf{R_w}$ connectives, respectively

Table 3.1: Definitions of $\mathcal{F}_\circ$ and $\Delta_\circ$ for the binary connectives (θ_1, θ_2 conjunctions of atomic formulas over AP, $F_1 \subseteq \mathcal{F}_1$, $F_2 \subseteq \mathcal{F}_2$, $Q_1' \subseteq Q_1$, $Q_2' \subseteq Q_2$, and f is a new acceptance condition not yet used in the application of another translation rule)

$\circ$	$\mathcal{F}_\circ$	$\Delta_\circ$
$\vee$	$\emptyset$	$\Big\{\langle q_I, \theta_1, \emptyset, Q_1'\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q_1'\rangle \in \Delta_1\Big\} \cup \Big\{\langle q_I, \theta_2, \emptyset, Q_2'\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\}$
$\wedge$	$\emptyset$	$\Big\{\big\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q_1' \cup Q_2'\big\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q_1'\rangle \in \Delta_1, \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\}$
$\mathsf{U_s}$	$\{f\}$	$\Big\{\big\langle q_I, \theta_1, \{f\}, Q_1' \cup \{q_I\}\big\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q_1'\rangle \in \Delta_1\Big\} \cup \Big\{\langle q_I, \theta_2, \emptyset, Q_2'\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\}$
$\mathsf{U_w}$	$\emptyset$	$\Big\{\big\langle q_I, \theta_1, \emptyset, Q_1' \cup \{q_I\}\big\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q_1'\rangle \in \Delta_1\Big\} \cup \Big\{\langle q_I, \theta_2, \emptyset, Q_2'\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\}$
$\mathsf{R_s}$	$\{f\}$	$\Big\{\big\langle q_I, \theta_2, \{f\}, Q_2' \cup \{q_I\}\big\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\} \cup \Big\{\big\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q_1' \cup Q_2'\big\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q_1'\rangle \in \Delta_1, \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\}$
$\mathsf{R_w}$	$\emptyset$	$\Big\{\big\langle q_I, \theta_2, \emptyset, Q_2' \cup \{q_I\}\big\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\} \cup \Big\{\big\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q_1' \cup Q_2'\big\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q_1'\rangle \in \Delta_1, \langle q_{I2}, \theta_2, F_2, Q_2'\rangle \in \Delta_2\Big\}$

Thus, the formula can be rewritten as the logically equivalent formula

$$\Big[\Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big)\wedge\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big)\vee\big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)\Big];$$

this formula is clearly in positive normal form. Because the main connective $\vee$ of the formula is a binary connective, we first have to find automata for the formulas $\varphi \overset{def}{=} \Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big)\wedge\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big)$ and $\psi \overset{def}{=} \big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)$ before we can apply the translation rule for the $\vee$ connective. Proceeding recursively towards the subformulas of the formula φ, we first build automata shown in Fig. 3.2 (a) for the subformulas $\bot$, $\top$, p_1 and p_2 of φ. We can then define automata for the subformulas $(\top\,\mathsf{U_s}\,p_1)$ and $(\top\,\mathsf{U_s}\,p_2)$ by applying the translation rule given for the $\mathsf{U_s}$ connective first to $\mathcal{A}_\top$ and $\mathcal{A}_{p_1}$, and then to $\mathcal{A}_\top$ and $\mathcal{A}_{p_2}$; see Fig. 3.2 (b). Because the subformulas are strong temporal eventualities, we associate a unique acceptance condition with each compound automaton. (To simplify these and the following figures, we shall always omit the states not reachable from the initial states of the constructed automata, since they can be removed from the automata by Proposition 2.3.12 without changing their languages.)

We then apply the $\mathsf{R_w}$ translation rule to the automata $\mathcal{A}_\bot$ and $\mathcal{A}_{(\top\,\mathsf{U_s}\,p_1)}$, and then to $\mathcal{A}_\bot$ and $\mathcal{A}_{(\top\,\mathsf{U_s}\,p_2)}$, to obtain the automata shown in Fig. 3.2 (c). Because $\mathsf{R_w}$ is a weak temporal eventuality, no new acceptance conditions are added to the automata. (Because the automaton $\mathcal{A}_\bot$ has no initial transitions, the initial transitions of $\mathcal{A}_{(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_i))}$ are completely determined by the automaton $\mathcal{A}_{(\top\,\mathsf{U_s}\,p_i)}$ for all $i \in \{1, 2\}$.)

We next merge the automata built for the top-level subformulas of φ into the automaton shown in Fig. 3.2 (d) for the formula φ itself by using the $\wedge$ translation rule.

The translation of the subformula $\big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)$ proceeds similarly. We start from the automata built for the atomic subformulas (see Fig. 3.3 (a)) and apply the $\mathsf{R_s}$ translation rule to $\mathcal{A}_{p_4}$ and $\mathcal{A}_{p_5}$ to obtain an automaton for the formula $(p_4\,\mathsf{R_s}\,p_5)$ (Fig. 3.3 (b)). Again, because $(p_4\,\mathsf{R_s}\,p_5)$ is a strong temporal eventuality, we add a new acceptance condition to the automaton. We then apply the $\mathsf{R_w}$ rule to $\mathcal{A}_{p_3}$ and $\mathcal{A}_{(p_4\,\mathsf{R_s}\,p_5)}$ to construct an automaton for the formula ψ (Fig. 3.3 (c)).

We finally apply the $\vee$ translation rule to $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ to build the automaton shown in Fig. 3.4 for the formula

$$\Big[\Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big)\wedge\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big)\vee\big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)\Big].$$

■

3.1.1 Simple Observations

Correspondence between node subformulas of φ and states of $\mathcal{A}_\varphi$. The construction of an automaton $\mathcal{A}_\varphi$ for the given LTL formula φ (in positive normal form) is guided by the structure of φ, which completely determines the set of rules that need be applied for translating the formula into an automaton. Even though the particular application order of the rules may remain partially unspecified (i.e., automata for any pair of subformulas of φ

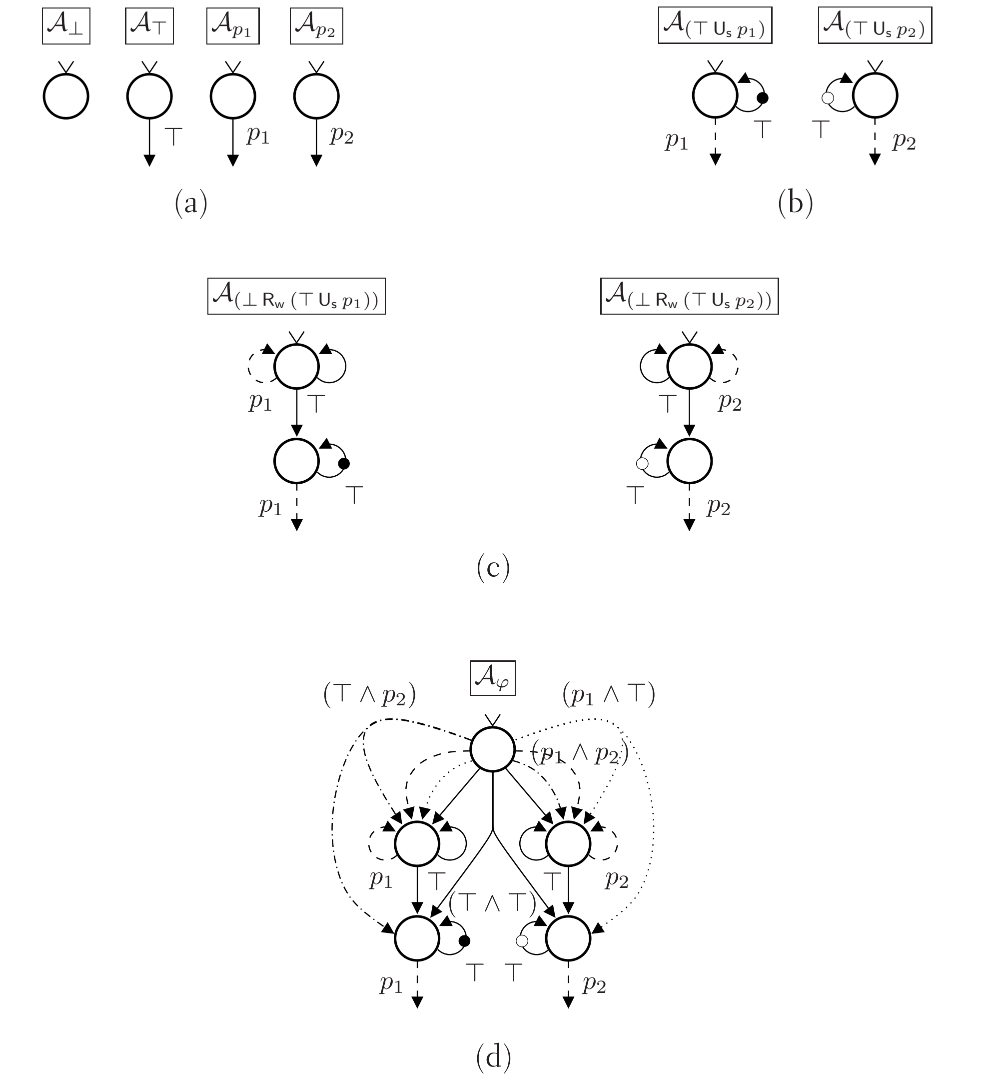


Fig. 3.2: Building an automaton for the LTL formula $\varphi \stackrel{def}{=} \Big(\big(\bot \mathsf{R_w} (\top \mathsf{U_s}\, p_1)\big) \wedge \big(\bot \mathsf{R_w} (\top \mathsf{U_s}\, p_2)\big)\Big)$. (a) Automata for the atomic subformulas of φ; (b) Automata for the formulas $(\top \mathsf{U_s}\, p_1)$ and $(\top \mathsf{U_s}\, p_2)$; (c) Automata for the formulas $\big(\bot \mathsf{R_w} (\top \mathsf{U_s}\, p_1)\big)$ and $\big(\bot \mathsf{R_w} (\top \mathsf{U_s}\, p_2)\big)$; (d) Automaton for the formula φ

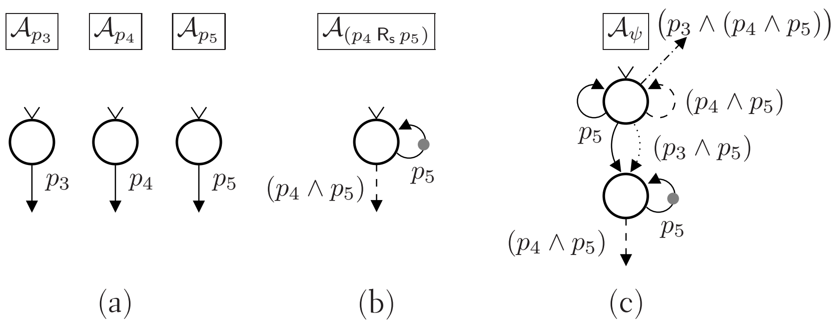


Fig. 3.3: Building an automaton for the formula $\psi \stackrel{def}{=} \big(p_3 \mathsf{R_w} (p_4 \mathsf{R_s}\, p_5)\big)$. (a) Automata for the atomic subformulas of ψ; (b) Automaton for the formula $(p_4 \mathsf{R_s}\, p_5)$; (c) Automaton for the formula ψ

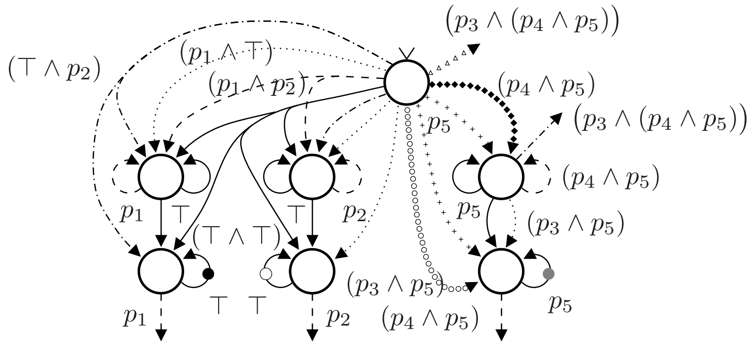


Fig. 3.4: Automaton for the LTL formula $\Big[\Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big)\land\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big)\lor \big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)\Big]$

that do not share any subformulas can be constructed in either order), automata built for two syntactically identical subformulas of φ are nevertheless easily seen to be isomorphic (i.e., one can be obtained from the other by renaming its states and acceptance conditions). It is therefore possible to reuse the structure of the automata constructed during the translation by directing the transitions added in the application of a translation rule to previously added states whenever possible. Because the recursive translation rules treat the atomic subformulas of φ as the base case, it follows that the number of rule applications required equals the number of node subformulas of φ. Therefore, because $\mathrm{NSub}(\varphi)$ is finite, the translation always terminates, and, because each step of the translation adds exactly one new state to the result, there is a bijective correspondence between $\mathrm{NSub}(\varphi)$ and the set of states in the automaton built for the formula φ.

Interpretation of the translation rules. The correspondence between $\mathrm{NSub}(\varphi)$ and the states of the automaton $\mathcal{A}_\varphi$ gives a simple interpretation of each translation rule. Intuitively, each translation rule gives instructions on how to recognize the language of an LTL formula φ by either giving an automaton for φ directly (the rules for the atomic formulas), or describing how to run the automata built for the top-level subformula(s) of a non-atomic formula φ to recognize the language $\mathcal{L}(\varphi)$. Thus, for example, the translation rule for constructing an automaton $\mathcal{A}_{(\varphi_1\land\varphi_2)}$ for the language $\mathcal{L}\big((\varphi_1 \land \varphi_2)\big)$ interprets to first building the automata $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ for the languages $\mathcal{L}(\varphi_1)$ and $\mathcal{L}(\varphi_2)$ and then creating an automaton that, in effect, runs $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ in parallel on any given input. The translation rule makes the first admissible transition in any run of $\mathcal{A}_{(\varphi_1\land\varphi_2)}$ mimic a pair of initial transitions taken synchronously by each of the component automata. As a result, the initial transition in any run of $\mathcal{A}_{(\varphi_1\land\varphi_2)}$ corresponds to spawning both $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ on the same input. However, since this initial transition already synchronizes by itself with the first symbol of the input, the target states of the transition need be adjusted so that the set of copies of the automaton which are active after the transition matches the copies of $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ that would be active after a synchronous pair of initial transitions. This is the reason for not including the initial states of the component automata in the target states of the initial transitions of $\mathcal{A}_{(\varphi_1 \wedge \varphi_2)}$ unless the transitions are self-loops, which are thus unrolled in the application of the translation rule.

A corresponding adjustment of target states is needed for defining the initial transitions of automata built for any other connective, except for the Next Time operator X; namely, the purpose of the Next Time translation rule is to modify an automaton built for an LTL formula φ into an automaton that, in effect, postpones the checking of φ by one initial step.

For the binary temporal connectives $\mathsf{U_s}$ and $\mathsf{U_w}$, the definition of $\Delta_\circ$ in the translation rules is a direct automata-based encoding of the well-known LTL identity

$$(\varphi_1 \,\mathsf{U}\, \varphi_2) \equiv \Big(\varphi_2 \vee \big(\varphi_1 \wedge \mathsf{X}(\varphi_1 \,\mathsf{U}\, \varphi_2)\big)\Big)$$

where U is an Until connective of the same strength on both sides of the identity. Thus, for example, the translation of the formula $(\varphi_1 \,\mathsf{U_s}\, \varphi_2)$ into an automaton corresponds to first building automata for the top-level subformulas and then joining them into an automaton that verifies that either φ_2 holds in the infinite suffix of the input beginning at the current input position, or that φ_1 holds in this suffix and $(\varphi_1 \,\mathsf{U_s}\, \varphi_2)$ still holds in the infinite suffix beginning at the next input position. In the latter case, the automaton spawns two independent copies of itself, one of which checks whether the infinite suffix beginning at the current input position belongs to $\mathcal{L}(\varphi_1)$, whereas the other proceeds to check whether $(\varphi_1 \,\mathsf{U_s}\, \varphi_2)$ still holds from the next input position onward.

The rules for the Release connectives can be derived from the rules introduced for the $\wedge$ and Until connectives via the identities

$$(\varphi_1 \,\mathsf{R_s}\, \varphi_2) \equiv \big(\varphi_2 \,\mathsf{U_s}\, (\varphi_1 \wedge \varphi_2)\big) \quad \text{and} \quad (\varphi_1 \,\mathsf{R_w}\, \varphi_2) \equiv \big(\varphi_2 \,\mathsf{U_w}\, (\varphi_1 \wedge \varphi_2)\big).$$

Formally, the combination of the rules introduced for the $\wedge$ and the Until connectives defines an automaton with a state that is unreachable from the initial state of the automaton, namely, the initial state of the automaton constructed for the formula $(\varphi_1 \wedge \varphi_2)$. However, this state can be safely discarded by Proposition 2.3.12 without changing the language of the automaton. This simplification then gives the direct rules shown in Table 3.1.

Loop structure. Building an automaton for a compound formula φ from one or two component automata constructed for the top-level subformula(s) of φ is done by taking a new initial state for the automaton and then adding transitions from this state to itself and the states of the component automata as instructed by the translation rules. Since no rule manipulates the transition relation of any component automaton, it follows (by induction) that every state q of the automaton $\mathcal{A}_\varphi$ constructed for an LTL formula φ will always remain unreachable from all of its descendants except possibly q itself. Thus, all loops in the transition structure of the final automaton will be self-loops, which may arise in the application of the translation rules to subformulas of φ with a binary temporal main connective. By these observations, it follows that the automaton constructed by the translation rules is a self-loop alternating automaton.

Structure of transition guards. The guard of the only transition in an automaton built for any atomic formula different from $\bot$ is simply the formula itself, encoding the subsets of AP that satisfy it: for $\top$, all subsets of AP; for literals, all subsets of AP that (for negative literals, do not) include the proposition in the literal. The guards of the initial transitions of any compound automaton are either $\top$ (the Next Time operator), or they are built from the guards of the initial transitions of the component automata. It is easy to see that each new transition either inherits its guard directly from another transition, or the guard is built as the conjunction of two previously defined guards. By induction, it follows that all guards in the final automaton will be finite conjunctions of one or more atomic formulas, corresponding to finite intersections of one or more subsets of 2^{AP} by the semantics of $\wedge$. This very restricted form allows for efficient checking of propositional implications between the guards, which is needed for the automaton simplification constructions discussed in Ch. 6.

Acceptance conditions. New acceptance conditions are introduced to the constructed automaton whenever applying one of the translation rules to a subformula having either of the strong binary temporal operators ($\mathsf{U_s}$ or $\mathsf{R_s}$) as its main connective. Intuitively, because the conditions are interpreted as fin-acceptance conditions, they will prevent the automaton from remaining in a state corresponding to an unsatisfied strong temporal eventuality indefinitely along any path through a fin-accepting run of the automaton. Therefore, the acceptance of an input requires the eventual satisfaction of each strong temporal eventuality along the input as required by the semantics of the strong temporal operators. This intuition will be made formal in the correctness proof of Sect. 3.3.

As seen from the translation rules, the transitions added to the automaton at each step never inherit any acceptance conditions from previously defined transitions. Since each translation rule adds at most one acceptance condition to the automaton, it follows that the set of acceptance conditions of each transition of the final automaton will be either an empty or a singleton set. Since all transitions with a nonempty set of acceptance conditions are self-loops of the automaton, the final automaton is easily seen to be constructed simplified in the sense of Corollary 2.3.20. Additionally, it is easy to see from the translation rules that all transitions of the final automaton having a particular acceptance condition in their set of acceptance conditions always have the same source state.[1]

[1]Actually, this fact can be used (together with Proposition 2.3.18) to show that it is not necessary to associate a unique acceptance condition with each strong temporal eventuality, i.e., all eventualities could share the same acceptance condition as in the translation of Gastin and Oddoux [2001]. We shall not do this here, however, since the correctness of many heuristics for improving the translation (to be presented in the following chapters) relies on the strict correspondence between acceptance conditions and temporal eventualities.

3.2 SIZES OF COMPONENTS IN AN AUTOMATON BUILT FROM AN LTL FORMULA

In this section we consider upper bounds for sizes of components of an automaton built from (the positive normal form) of an LTL formula $\varphi \in LTL(AP)$. The sizes of the components of the subautomaton $\mathcal{A}_\varphi^{q_I}$ rooted at the initial state of the automaton $\mathcal{A}_\varphi$ (built from $[\varphi]^{\mathrm{PNF}}$ using the translation rules of Sect. 3.1) satisfy the inequalities

- $|Q| \leq 1 + \left|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\right| \leq 1 + 2 \cdot |\mathrm{Temp}(\varphi)|$ (Sect. 3.2.1),
- $|\Delta| \leq 2^{\mathrm{NSize}([\varphi]^{\mathrm{PNF}})-1} < 2^{2\cdot|\varphi|-1}$ (Sect. 3.2.2), and
- $|\mathcal{F}| \leq \left|\big\{(\varphi_1 \circ \varphi_2) \in \mathrm{Sub}([\varphi]^{\mathrm{PNF}}) : \circ \in \{\mathsf{U_s}, \mathsf{R_s}\}\big\}\right|$. (Sect. 3.2.3)

3.2.1 Number of States

As noted previously, an LTL formula φ (in positive normal form) can be translated into an automaton in $|\mathrm{NSub}(\varphi)|$ applications of a translation rule. Since the rules build the automaton one state at a time, the translation ends after exactly $|\mathrm{NSub}(\varphi)|$ states have been defined. Therefore, the size of $\mathrm{NSub}(\varphi)$ also gives a simple upper bound for the number of states in an automaton recognizing the language of the formula φ.

As seen already in Ex. 3.1.1, the application of a translation rule to define an alternating automaton from smaller component automata may leave some states in the component automata unreachable from the initial state of the newly constructed automaton. However, this fact is not taken into account when using the number of translation steps as an upper bound for the number of states in an automaton $\mathcal{A}_\varphi$ built for a given LTL formula. Because the language of an alternating automaton depends only on those states of the automaton that are actually reachable from the initial state of the automaton (Proposition 2.3.12), a tighter bound can be given by considering the number of states in the subautomaton $\mathcal{A}_\varphi^{q_I}$ obtained from the result of the translation by restricting it to the smallest set of states that includes the state q_I and the states actually reachable from q_I. For this purpose, we examine the translation rules to find the exact conditions under which a state introduced during the translation will still be reachable from the initial state of the final automaton.

Each translation rule for building a compound automaton either adds a transition to an initial state of a component automaton (the Next Time rule), or it uses the initial transitions of the component automata as a basis for defining the initial transitions of the compound automaton (rules for the binary connectives). It is clear from the translation rules that all target states of each initial transition of a component automaton will be included as target states of some transition of the compound automaton. Additionally, since none of the rules ever change—or even refer to—the non-initial transitions of any component automaton, it follows that a state reachable from the initial state of a component automaton will remain reachable from the initial state of any automaton obtained from it by any number of translation rules. By examining the translation rules, we find that the initial state q_I of some component automaton will still be included in the subautomaton rooted at the initial state of the final automaton at least if it satisfies one of the following conditions:

- q_I is the initial state of the final automaton built for the LTL formula φ. Clearly, because q_I is the last state to be added into the automaton, the final automaton is never used as a component automaton in any translation rule.
- q_I has a self-loop transition to itself, which is possible (by the definition of the translation rules) only if q_I is the initial state of an automaton built for a binary pure temporal subformula (i.e., a subformula with either $\mathsf{U_s}$, $\mathsf{U_w}$, $\mathsf{R_s}$ or $\mathsf{R_w}$ as its main connective).
- q_I is the initial state of an automaton corresponding to a subformula φ_1, and $\mathsf{X}\varphi_1 \in \mathrm{NSub}(\varphi)$. (Since $\mathsf{X}\varphi_1 \in \mathrm{NSub}(\varphi)$, the Next Time rule will be applied to the automaton $\mathcal{A}_{\varphi_1}$ in the translation; the application of the rule then results in an automaton with an initial transition to q_I.)

We show that the three above conditions actually describe the *exact* set of states in the subautomaton rooted at the initial state of the final automaton. Assume that q_I is the initial state of an automaton (corresponding to a formula $\varphi_1 \in \mathrm{NSub}(\varphi)$) such that q_I satisfies none of the above conditions. Then, φ has a non-atomic compound subformula with φ_1 as a top-level subformula. Because $\mathsf{X}\varphi_1 \notin \mathrm{NSub}(\varphi)$, all such subformulas are binary subformulas of φ. Let φ' be any of these formulas. When a translation rule is applied to construct the automaton $\mathcal{A}_{\varphi'}$, the state q_I will not be connected to the initial state of $\mathcal{A}_{\varphi'}$, because q_I has no self-loop transitions. Because $\mathsf{X}\varphi_1 \notin \mathrm{NSub}(\varphi)$, it follows that q_I cannot be connected to the initial state of another automaton constructed later in the procedure, and thus q_I will remain unreachable from the initial state of the final automaton. We have thus proved the following result:

Proposition 3.2.1 *Let $\mathcal{A}_\varphi$ be the alternating automaton built for the LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ using the translation rules, and let $\mathcal{A}_\varphi^{q_I}$ (with state set Q) be the subautomaton rooted at the initial state of $\mathcal{A}_\varphi$. Then,*

$$|Q| = \left| \begin{array}{l} \{\varphi\} \\ \cup \big\{(\varphi_1 \circ \varphi_2) \in \mathrm{NSub}(\varphi) : \circ \in \{\mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}\big\} \\ \cup \big\{\varphi_1 \in \mathrm{NSub}(\varphi) : \mathsf{X}\varphi_1 \in \mathrm{NSub}(\varphi)\big\} \end{array} \right|$$

This result leads to the following upper bound for the number of states in an alternating automaton constructed from any LTL formula (that is not necessarily in positive normal form). The upper bound is essentially the same as the one that is implicit in the translation of Gastin and Oddoux [2001].

Corollary 3.2.2 *Let $\varphi \in LTL(AP)$ be any LTL formula built from the elements of AP, the Boolean constants $\top$ and $\bot$, and the connectives $\{\neg, \vee, \wedge, \mathsf{X}, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$. The language of the formula φ can be recognized by an alternating automaton on the alphabet 2^{AP} with at most $1 + \big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big| \leq 1 + 2 \cdot |\mathrm{Temp}(\varphi)|$ states ($1 + |\mathrm{Temp}(\varphi)|$ states, if φ itself is in positive normal form). (If φ is a binary pure temporal formula, the upper bound reduces to $\big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big|$ states.)*

Proof: As noted in Sect. 2.2.3, the formula $[\varphi]^{\mathrm{PNF}}$ (which is in positive normal form) has at most twice as many pure temporal subformulas as φ, i.e., $\big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big| \leq 2 \cdot |\mathrm{Temp}(\varphi)|$, and $\mathcal{L}\big([\varphi]^{\mathrm{PNF}}\big) = \mathcal{L}(\varphi)$. By applying the translation to $[\varphi]^{\mathrm{PNF}}$, the result follows directly from Proposition 3.2.1 by observing that

$$\left|\begin{array}{l}\big\{(\varphi_1 \circ \varphi_2) \in \mathrm{NSub}\big([\varphi]^{\mathrm{PNF}}\big) : \circ \in \{\mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}\big\} \\ \cup \big\{\varphi_1 \in \mathrm{NSub}\big([\varphi]^{\mathrm{PNF}}\big) : \mathsf{X}\varphi_1 \in \mathrm{NSub}\big([\varphi]^{\mathrm{PNF}}\big)\big\}\end{array}\right| \leq \big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big|.$$

□

3.2.2 Number of Transitions

By Corollary 3.2.2, any LTL formula can be translated into an alternating automaton with a linear number of states in the number of pure temporal subformulas in the formula. However, it is not difficult to see that there is a price to pay for the explicit representation of transitions: an automaton built using the translation rules may have exponentially many transitions in the length of the formula in the worst case. First, it is easy to show that the number of transitions defined in the translation of (the positive normal form of) an LTL formula $\varphi \in LTL(AP)$ into an automaton is exponentially bounded by $\mathrm{NSize}\big([\varphi]^{\mathrm{PNF}}\big)$:

Proposition 3.2.3 *Let $\mathcal{A}_\varphi$ be an alternating automaton built from (the positive normal form of) an LTL formula $\varphi \in LTL(AP)$ using the translation rules presented in Sect. 3.1. The automaton $\mathcal{A}_\varphi$ has at most $2^{\mathrm{NSize}([\varphi]^{\mathrm{PNF}})-1} < 2^{2\cdot|\varphi|-1}$ transitions.*

Proof: We first prove the result for formulas in positive normal form. Let $\varphi \in LTL^{\mathrm{PNF}}(AP)$, and let Δ be the set of transitions of $\mathcal{A}_\varphi$. If $\mathrm{NSize}(\varphi) = 1$, then φ is an atomic formula. By the translation rules, Δ contains at most one element, and because $2^{\mathrm{NSize}(\varphi)-1} = 2^0 = 1$, the result holds in this case.

Assume that the result holds for all LTL formulas whose node size is less than or equal to some fixed $1 \leq k < \omega$, and let φ be a compound formula with node size $k+1$. Then, $\varphi = \mathsf{X}\varphi_1$ or $\varphi = (\varphi_1 \circ \varphi_2)$ for some $\circ \in \{\vee, \wedge, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$ and $\varphi_1, \varphi_2 \in LTL^{\mathrm{PNF}}(AP)$ such that $\mathrm{NSize}(\varphi_1) \leq k$ and $\mathrm{NSize}(\varphi_2) \leq k$ hold. Let $\mathcal{A}_1 = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$ and $\mathcal{A}_2 = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2\rangle$ be the automata built using the translation rules for the formulas φ_1 and φ_2, respectively. Additionally, let $\Delta_1^{q_{I1}} \subseteq \Delta_1$ and $\Delta_2^{q_{I2}} \subseteq \Delta_2$ denote the initial transitions of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively (i.e., $\Delta_i^{q_{Ii}} \stackrel{def}{=} \big\{\langle q, \theta, F, Q'\rangle \in \Delta_i \,\big|\, q = q_{Ii}\big\}$ for all $i \in \{1, 2\}$). There are the following cases:

$(\varphi = \mathsf{X}\varphi_1)$

$$\begin{aligned} |\Delta| &= |\Delta_1| + 1 && \textit{(translation rule for the } \mathsf{X} \textit{ connective)} \\ &\leq 2^{\mathrm{NSize}(\varphi_1)-1} + 1 && \textit{(induction hypothesis)} \\ &\leq 2^{\mathrm{NSize}(\varphi_1)-1} + 2^{\mathrm{NSize}(\varphi_1)-1} && (\mathrm{NSize}(\varphi_1) \geq 1) \\ &= 2^{\mathrm{NSize}(\varphi_1)} \\ &= 2^{\mathrm{NSize}(\varphi)-1} && \textit{(definition of } \mathrm{NSize}(\varphi)) \end{aligned}$$

$\underline{(\varphi = (\varphi_1 \circ \varphi_2), \circ \in \{\vee, \mathsf{U_s}, \mathsf{U_w}\})}$

$$
\begin{aligned}
|\Delta| &\le |\Delta_1| + |\Delta_2| + |\Delta_1^{q_{I1}}| + |\Delta_2^{q_{I2}}| && \textit{(translation rules)}\\
&\le |\Delta_1| + |\Delta_2| + |\Delta_1| + |\Delta_2| \\
&= 2 \cdot \big(|\Delta_1| + |\Delta_2|\big) \\
&\le 2 \cdot \big(2^{\mathrm{NSize}(\varphi_1)-1} + 2^{\mathrm{NSize}(\varphi_2)-1}\big) && \textit{(induction hypothesis)}\\
&= 2^{\mathrm{NSize}(\varphi_1)} + 2^{\mathrm{NSize}(\varphi_2)} \\
&\le 2^{\mathrm{NSize}(\varphi_1)} \cdot 2^{\mathrm{NSize}(\varphi_2)} && (\mathrm{NSize}(\varphi_1), \mathrm{NSize}(\varphi_2) \ge 1)\\
&= 2^{\mathrm{NSize}(\varphi_1)+\mathrm{NSize}(\varphi_2)} \\
&= 2^{\mathrm{NSize}(\varphi)-1} && \textit{(definition of } \mathrm{NSize}(\varphi))
\end{aligned}
$$

$\underline{(\varphi = (\varphi_1 \wedge \varphi_2))}$

$$
\begin{aligned}
|\Delta| &\le |\Delta_1| + |\Delta_2| + |\Delta_1^{q_{I1}}| \cdot |\Delta_2^{q_{I2}}| && \textit{(translation rules)}\\
&\le |\Delta_1| + |\Delta_2| + |\Delta_1| \cdot |\Delta_2| \\
&\le 2^{\mathrm{NSize}(\varphi_1)-1} + 2^{\mathrm{NSize}(\varphi_2)-1} + 2^{\mathrm{NSize}(\varphi_1)-1} \cdot 2^{\mathrm{NSize}(\varphi_2)-1} \\
& && \textit{(induction hypothesis)}\\
&\le 2^2 \cdot 2^{\mathrm{NSize}(\varphi_1)-1} \cdot 2^{\mathrm{NSize}(\varphi_2)-1} && (\mathrm{NSize}(\varphi_1), \mathrm{NSize}(\varphi_2) \ge 1)\\
&= 2^{\mathrm{NSize}(\varphi_1)+\mathrm{NSize}(\varphi_2)} \\
&= 2^{\mathrm{NSize}(\varphi)-1} && \textit{(definition of } \mathrm{NSize}(\varphi))
\end{aligned}
$$

$\underline{(\varphi = (\varphi_1 \circ \varphi_2), \circ \in \{\mathsf{R_s}, \mathsf{R_w}\})}$

$$
\begin{aligned}
|\Delta| &\le |\Delta_1| + |\Delta_2| + |\Delta_2^{q_{I2}}| + |\Delta_1^{q_{I1}}| \cdot |\Delta_2^{q_{I2}}| && \textit{(translation rules)}\\
&\le |\Delta_1| + |\Delta_2| + |\Delta_2| + |\Delta_1| \cdot |\Delta_2| \\
&\le 2^{\mathrm{NSize}(\varphi_1)-1} + 2 \cdot 2^{\mathrm{NSize}(\varphi_2)-1} + 2^{\mathrm{NSize}(\varphi_1)-1} \cdot 2^{\mathrm{NSize}(\varphi_2)-1} \\
& && \textit{(induction hypothesis)}\\
&\le 2^2 \cdot 2^{\mathrm{NSize}(\varphi_1)-1} \cdot 2^{\mathrm{NSize}(\varphi_2)-1} && (\mathrm{NSize}(\varphi_1), \mathrm{NSize}(\varphi_2) \ge 1)\\
&= 2^{\mathrm{NSize}(\varphi_1)+\mathrm{NSize}(\varphi_2)} \\
&= 2^{\mathrm{NSize}(\varphi)-1} && \textit{(definition of } \mathrm{NSize}(\varphi))
\end{aligned}
$$

By induction on $|\varphi|$, it follows that the automaton built from any formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ using the translation rules has at most $2^{\mathrm{NSize}(\varphi)-1}$ transitions. If $\varphi \in LTL(AP)$ is not in positive normal form, then the result follows because $\mathrm{NSize}([\varphi]^{\mathrm{PNF}}) < 2 \cdot |\varphi|$ holds. $\square$

On the other hand, for every $1 \le n < \omega$, it is possible to find an LTL formula φ such that $\mathrm{NSize}(\varphi) \in O(n)$ holds, but an automaton built from φ using the translation rules has $2^{O(n)}$ transitions (even when restricted to the subautomaton rooted at its initial state).

Example 3.2.4 Let $\{\varphi_n\}_{1 \le n < \omega}$ (where φ_n is an LTL formula over n atomic propositions $\{p_1, p_2, \ldots, p_n\}$ for all $1 \le n < \omega$) be a set of LTL formulas defined inductively as

$$
\begin{aligned}
\varphi_1 &\overset{def}{=} (\top \,\mathsf{U_s}\, p_1), \quad \text{and} \\
\varphi_{n+1} &\overset{def}{=} \big(\varphi_n \wedge (\top \,\mathsf{U_s}\, p_{n+1})\big) \quad \text{for all } 1 \le n < \omega.
\end{aligned}
$$

Clearly, φ_n is in positive normal form for all $1 \le n < \omega$, and $\mathrm{NSize}(\varphi_n) = 4n - 1 \in O(n)$ holds for all $1 \le n < \omega$ ($\mathrm{NSize}(\varphi_1) = \mathrm{NSize}\big((\top \,\mathsf{U_s}\, p_1)\big) = 3 = 4 \cdot 1 - 1$, and if $\mathrm{NSize}(\varphi_k) = 4k - 1$ holds for some $1 \le k < \omega$, then $\mathrm{NSize}(\varphi_{k+1}) = 1 + \mathrm{NSize}(\varphi_k) + \mathrm{NSize}\big((\top \,\mathsf{U_s}\, p_{k+1})\big) = 1 + (4k-1) + 3 = 4(k+1) - 1$).

Let q_{I_n} be the initial state of $\mathcal{A}_{\varphi_n}$. By Proposition 3.2.3, $\mathcal{A}_{\varphi_n}$ has at most $2^{\mathrm{NSize}(\varphi_n)-1} \in 2^{O(n)}$ transitions. We show that $\mathcal{A}_{\varphi_n}$ has at least 2^n initial transitions. It is easy to see that the automaton built for any of the subformulas of the form $(\top \,\mathsf{U_s}\, p_n)$ using the translation rules has two initial transitions. If $\mathcal{A}_{\varphi_k}$ has 2^k initial transitions for some $1 \leq k < \omega$, then $\mathcal{A}_{\varphi_{k+1}}$ has $2^k \cdot 2 = 2^{k+1}$ initial transitions by the definition of the translation rule given for the $\wedge$ connective. Obviously, these transitions remain in the subautomaton rooted at the initial state of $\mathcal{A}_{\varphi_{k+1}}$, and it follows that $\mathcal{A}_{\varphi_n}^{q_{I_n}}$ has $2^{O(n)}$ transitions for all $1 \leq n < \omega$. ■

From the above example it follows that the result of the translation procedure from LTL to self-loop alternating automata based on the rules presented in Sect. 3.1 may need exponential space in the node size of the given formula in the worst case. Clearly, the translation will in such cases require also at least exponential time in the node size of the formula. This worst-case behavior is caused by the cumulative effect of applying translation rules defined for the $\wedge$ and R connectives to LTL formulas containing nested occurrences of these connectives: to define the initial transitions in a compound automaton built using one of these rules, it is always necessary to enumerate all pairwise combinations of initial transitions of the component automata.

3.2.3 Number of Acceptance Conditions

Let $\mathcal{A}_\varphi = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ be a self-loop alternating automaton built for the positive normal form of an LTL formula $\varphi \in LTL(AP)$ using the translation rules. As noted already at the end of Sect. 3.1, new acceptance conditions are introduced during the translation whenever applying a translation rule to a subformula corresponding to a strong temporal eventuality (i.e., a formula with $\mathsf{U_s}$ or $\mathsf{R_s}$ as its main connective), and thus

$$|\mathcal{F}| \leq \Big|\big\{(\varphi_1 \circ \varphi_2) \in \mathrm{Sub}([\varphi]^{\mathrm{PNF}}) : \circ \in \{\mathsf{U_s}, \mathsf{R_s}\}\big\}\Big|.$$

3.3 CORRECTNESS OF THE TRANSLATION

In this section we show the correctness of the translation. We start by proving a lemma that characterizes fin-acceptance in a self-loop alternating automaton built using the translation rules for an LTL formula having $\mathsf{U_s}$ or $\mathsf{U_w}$ as its main connective and establishes a direct correspondence between the semantics of LTL and the behavior of these automata.

Lemma 3.3.1 *Let $\varphi = (\varphi_1 \circ \varphi_2) \in LTL^{\mathrm{PNF}}(AP)$ ($\circ \in \{\mathsf{U_s}, \mathsf{U_w}\}$), and let $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$, $\mathcal{A}_1 = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$ and $\mathcal{A}_2 = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2\rangle$ be the self-loop alternating automata constructed using the translation rules for φ, φ_1 and φ_2, respectively. For all $w \in (2^{AP})^\omega$,*

$\mathcal{A}$ fin-accepts w iff there exists an index $0 \leq i < \omega$ such that $\mathcal{A}_2$ fin-accepts w^i, and for all $0 \leq j < i$, $\mathcal{A}_1$ fin-accepts w^j
or
$\circ = \mathsf{U_w}$, and $\mathcal{A}_1$ fin-accepts w^i for all $0 \leq i < \omega$.

Proof: (Only if) Assume that $\mathcal{A}$ fin-accepts $w \in (2^{AP})^\omega$. Then, $\mathcal{A}$ has a fin-accepting run $G = \langle V, E, L\rangle$ on w. By Proposition 2.3.7, there exists an index $0 \leq i \leq \omega$ and a chain of edges $(e_j)_{0 \leq j < i+1}$, $e_j = \langle v_j, V'_j \rangle \in E \cap (V_j \times 2^{V_{j+1}})$, such that $L(e_j)$ is an initial self-loop of $\mathcal{A}$ for all $0 \leq j < i$, and if $i < \omega$, then $L(e_i)$ is an initial transition of $\mathcal{A}$ that is not a self-loop.

Because $L(e_j)$ is an initial self-loop of $\mathcal{A}$ for all $0 \leq j < i$, it follows from the translation rules that for each such self-loop there exists a corresponding initial transition of $\mathcal{A}_1$ for all $0 \leq j < i$. Furthermore, if $i < \omega$, the transition $L(e_i)$ (which is not a self-loop of $\mathcal{A}$) corresponds to some initial transition of $\mathcal{A}_2$. Let $0 \leq j < i+1$, and let $L(e_j) = L(\langle v_j, V'_j \rangle) = t = \langle q_I, \theta, F, Q' \rangle \in \Delta$ for some $\theta \in PL(AP)$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$. Because G is a run, $w(j) \models \theta$ and $Q' = L(V'_j)$ hold. We consider the above two cases separately.

- If t is a self-loop of $\mathcal{A}$, there exists a transition $\langle q_{I1}, \theta, F_1, Q'_1 \rangle \in \Delta_1$ for some $F_1 \subseteq \mathcal{F}_1$ and $Q'_1 \subseteq Q_1$ such that $Q' = Q'_1 \cup \{q_I\}$ holds.

  Because G is fin-accepting, each subautomaton $\mathcal{A}^{L(v')}$ has a fin-accepting run on $w^{j+1} = (w^j)^1$ for all $v' \in V'_j$ by Proposition 2.3.9, and because $Q'_1 \subseteq Q' = L(V'_j)$ holds and $\mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q',\mathcal{F}_1}) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q'})$ holds for all $q' \in Q_1$, $\mathcal{A}_1^{q'}$ has a fin-accepting run on w^{j+1} for all $q' \in Q'_1$. Moreover, because $w(j) = (w^j)(0) \models \theta$ holds, Proposition 2.3.15 shows that $\mathcal{A}_1$ has a fin-accepting run on w^j.

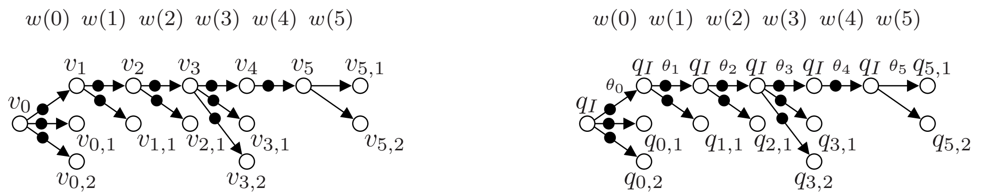
- If t is not a self-loop of $\mathcal{A}$, then t corresponds to an initial transition $\langle q_{I2}, \theta, F_2, Q' \rangle \in \Delta_2$ of $\mathcal{A}_2$ for some $F_2 \subseteq \mathcal{F}_2$, and thus $Q' \subseteq Q_2$ holds. Similarly to the self-loop case, the subautomaton $\mathcal{A}^{L(v')}$, which is fin-equivalent to $\mathcal{A}_2^{L(v')}$, fin-accepts w^{j+1} for all $v' \in V'_j$ by Proposition 2.3.9. Because $Q' = L(V'_j)$ and $w(j) \models \theta$ hold, it follows that $\mathcal{A}_2$ fin-accepts w^j (Proposition 2.3.15).

Thus, because $L(e_j)$ is a self-loop of $\mathcal{A}$ for all $0 \leq j < i$, it follows that $\mathcal{A}_1$ fin-accepts w^j for all $0 \leq j < i$, and furthermore, if $i < \omega$, then $\mathcal{A}_2$ fin-accepts w^i by the above discussion. It remains to show that the case $i = \omega$ is impossible if $\circ = \mathsf{U_s}$. If this were the case, then $\beta = (e_j)_{0 \leq j < i}$ would be an infinite branch in G having all of its edges labeled with initial self-loops of $\mathcal{A}$. However, because all of these self-loops share a common acceptance condition if $\circ = \mathsf{U_s}$, $\text{fin}(\beta)$ would be nonempty, which would contradict the assumption that G is a fin-accepting run of $\mathcal{A}$ on w. Therefore, if $\circ = \mathsf{U_s}$, then $i < \omega$ holds, and the result follows.

(If) Assume that there either exists an index $0 \leq i < \omega$ such that $\mathcal{A}_2$ fin-accepts w^i and for all $0 \leq j < i$, $\mathcal{A}_1$ fin-accepts w^j, or that $\circ = \mathsf{U_w}$ holds, and $\mathcal{A}_1$ fin-accepts w^i for all $0 \leq i < \omega$. That is, assume that there exists an index $0 \leq i \leq \omega$ such that $\mathcal{A}_1$ fin-accepts w^j for all $0 \leq j < i$, and if $i < \omega$, then $\mathcal{A}_2$ fin-accepts w^i.

By Proposition 2.3.15, the automaton $\mathcal{A}_1$ has an initial transition $t_{j,1} = \langle q_{I1}, \theta_{j,1}, F_{j,1}, Q'_{j,1} \rangle \in \Delta_1$ for some $\theta_{j,1} \in PL(AP)$, $F_{j,1} \subseteq \mathcal{F}_1$ and $Q'_{j,1} \subseteq Q_1$ for all $0 \leq j < i$ such that $w(j) \models \theta_{j,1}$ holds, and $\mathcal{A}_1^{q'}$ fin-accepts w^{j+1} for all $q' \in Q'_{j,1}$. Additionally, if $i < \omega$ holds, an analogous result holds for an initial transition $t_{i,2} = \langle q_{I2}, \theta_{i,2}, F_{i,2}, Q'_{i,2} \rangle \in \Delta_2$ of $\mathcal{A}_2$.

By the definition of $\mathcal{A}$, there now exists a transition $t_j = \langle q_I, \theta_j, F_j, Q'_j \rangle \in \Delta$, where $\theta_j = \theta_{j,1}$, $F_j = \{f\}$ for some new acceptance condition f ($\circ = \mathsf{U_s}$)



(a) (b)

Fig. 3.5: Construction of a semi-run in Lemma 3.3.1 ($\circ = \mathsf{U_s}$, $i = 5$). (a) Node and edge structure; (b) Labeling of nodes and edges

or $F_j = \emptyset$ ($\circ = \mathsf{U_w}$), and $Q'_j = Q'_{j,1} \cup \{q_I\}$ for all $0 \leq j < i$. Furthermore, if $i < \omega$ holds, then there exists also a transition $t_i = \langle q_I, \theta_i, \emptyset, Q'_i\rangle \in \Delta$, where $\theta_i = \theta_{i,2}$ and $Q'_i = Q'_{i,2}$. It is easy to see that, for all $0 \leq j < i+1$ and $q' \in Q'_j \setminus \{q_I\}$, $w(j) \models \theta_j$ holds, and $\mathcal{A}^{q'}$ fin-accepts w^{j+1} (because $Q'_j \setminus \{q_I\} = Q'_{j,1} \subseteq Q_1$ and $w^{j+1} \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q'}) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q',\mathcal{F}_1\cup\mathcal{F}}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ hold for all $0 \leq j < i$ and $q' \in Q'_{j,1}$ by the definition of $\mathcal{A}$; an analogous result holds for $Q'_{i,2}$ and the automaton $\mathcal{A}_2$ if $i < \omega$ holds). Using the transitions t_j, we define a fin-accepting semi-run of $\mathcal{A}$ on w.

(Definition of G) Write $Q'_j \setminus \{q_I\} = \{q_{j,1}, q_{j,2}, \ldots, q_{j,n_j}\} \subseteq Q$ for all $0 \leq j < i+1$ ($0 \leq n_j < \omega$, $q_{j,k} \neq q_{j,\ell}$ for all $1 \leq k, \ell \leq n_j$, $k \neq \ell$). Define the graph $G = \langle V, E, L\rangle$, where

- $V_0 \overset{def}{=} \{v_0\}$, $V_{j+1} \overset{def}{=} \{v_{j+1}, v_{j,1}, \ldots, v_{j,n_j}\}$ for all $0 \leq j < i$, and if $i < \omega$, let $V_{i+1} \overset{def}{=} \{v_{i,1}, \ldots, v_{i,n_i}\}$ and $V_j \overset{def}{=} \emptyset$ for all $i+1 < j < \omega$;
- $E \overset{def}{=} \bigcup_{0\leq j<i+1} \{\langle v_j, V_{j+1}\rangle\}$;
- $L(v_j) \overset{def}{=} q_I$, $L(v_{j,k}) \overset{def}{=} q_{j,k}$, and $L(\langle v_j, V_{j+1}\rangle) \overset{def}{=} t_j$ for all $0 \leq j < i+1$ and $1 \leq k \leq n_j$.

Figure 3.5 illustrates a possible structure for G with $\circ = \mathsf{U_s}$ and $i = 5$.

(G is a fin-accepting semi-run of $\mathcal{A}$ on w) We check that G is a fin-accepting semi-run of $\mathcal{A}$ on w.

(Partitioning) $V_0 = \{v_0\}$, and V is partitioned into finite disjoint levels (with edges only between successive levels) by construction.

(Causality) Let $v \in V_j$ for some $0 \leq j < \omega$. Then v either has no outgoing edges, or $v = v_j$. In this case v_j has the unique outgoing edge $e = \langle v_j, V_{j+1}\rangle \in E$. On the other hand, if $v \in V_j$ for some $1 \leq j < \omega$, then v is a successor of the node $v_{j-1} \in V_{j-1}$. It follows that G satisfies both the forward semi-causality and the backward causality constraints.

(Consistency of L) Clearly, $L(v_0) = q_I$ holds. Let $e \in E$. By construction, $e = \langle v_j, V_{j+1}\rangle$ holds for some $0 \leq j < i+1$. Because $L(e) = t_j = \langle q_I, \theta_j, F_j, Q'_j\rangle = \langle L(v_j), \theta_j, F_j, L(V_{j+1})\rangle \in \Delta$ and $w(j) \models \theta_j$ hold, it follows that the labeling L is consistent.

(Acceptance) If $i < \omega$, then the edge set E is finite. Therefore $\mathcal{B}(G) = \emptyset$, and G is trivially fin-accepting. Otherwise G contains a unique infinite branch, all edges in which are labeled with initial self-loops of $\mathcal{A}$. Because

$i = \omega$ implies that $\circ = \mathsf{U_w}$, the set of acceptance conditions of each initial self-loop of $\mathcal{A}$ is empty. It follows that the branch is fin-accepting, and thus G is a fin-accepting semi-run of $\mathcal{A}$ on w.

(G can be extended into a fin-accepting run) Let $v \in V$ be a node in G with no outgoing edges. Then $v = v_{j,k} \in V_{j+1}$ holds for some $0 \leq j < i+1$ and $1 \leq k \leq n_j$. Because $L(v) \in Q'_j \setminus \{q_I\}$ holds, and because $\mathcal{A}^{q'}$ fin-accepts w^j for all $q' \in Q'_j \setminus \{q_I\}$, it follows that G can be extended into a fin-accepting run of $\mathcal{A}$ on w by Proposition 2.3.14, and thus $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds. □

Using the above lemma together with Proposition 2.3.15, we can now give a simple inductive proof of the correctness of the translation.

Theorem 3.3.2 *Let $\varphi \in LTL^{\mathrm{PNF}}(AP)$ be an LTL formula in positive normal form, and let $\mathcal{A}_\varphi$ be an automaton constructed from φ using the translation rules. For all $w \in (2^{AP})^\omega$, $\mathcal{A}_\varphi$ fin-accepts w iff $w \models \varphi$ holds.*

Proof: We proceed by induction on the node size of the formula φ. If $\mathrm{NSize}(\varphi) = 1$, φ is an atomic formula. If $\varphi = \bot$, then it is easy to see from the definition of $\mathcal{A}_\varphi$ that $\mathcal{A}_\varphi$ has no runs on any input, and thus $\mathcal{L}_{\text{fin}}(\mathcal{A}_\varphi) = \emptyset = \mathcal{L}(\bot)$. Otherwise $\mathcal{A}_\varphi$ has exactly one initial transition (with guard φ), and it follows from Proposition 2.3.15 that $\mathcal{A}_\varphi$ has a fin-accepting run on w iff $w(0) \models \varphi$ holds, which is equivalent to $w \models \varphi$ in this case, because φ is a Boolean formula.

Assume that the result holds for all LTL formulas (in positive normal form) of node size less than or equal to some fixed $1 \leq k < \omega$, and let φ be a non-atomic LTL formula in positive normal form such that $\mathrm{NSize}(\varphi) = k+1$. We split the proof in separate cases based on the main connective of φ:

($\varphi = \mathsf{X}\varphi_1$)

$\mathcal{A}_\varphi$ fin-accepts w

iff there exists a transition $\langle q_I, \theta, F, Q' \rangle \in \Delta$ such that $w(0) \models \theta$ and for all $q \in Q'$, $\mathcal{A}^q_\varphi$ fin-accepts w^1 *(Proposition 2.3.15)*

iff $w(0) \models \top$ and $\mathcal{A}_{\varphi_1}$ fin-accepts w^1 *(definition of $\mathcal{A}_\varphi$)*

iff $w(0) \models \top$ and $w^1 \models \varphi_1$ *(induction hypothesis)*

iff $w \models \mathsf{X}\varphi_1$ *(semantics of LTL)*

($\varphi = (\varphi_1 \vee \varphi_2)$)

$\mathcal{A}_\varphi$ fin-accepts w

iff there exists a transition $\langle q_I, \theta, F, Q' \rangle \in \Delta$ such that $w(0) \models \theta$ and for all $q \in Q'$, $\mathcal{A}^q_\varphi$ fin-accepts w^1 *(Proposition 2.3.15)*

iff there exists a transition $\langle q_{I1}, \theta_1, F_1, Q'_1 \rangle \in \Delta_1$ such that $w(0) \models \theta_1$ and for all $q \in Q'_1$, $\mathcal{A}^q_\varphi$ $(= \mathcal{A}^q_{\varphi_1})$ fin-accepts w^1

or

there exists a transition $\langle q_{I2}, \theta_2, F_2, Q'_2 \rangle \in \Delta_2$ such that $w(0) \models \theta_2$ and for all $q \in Q'_2$, $\mathcal{A}^q_\varphi$ $(= \mathcal{A}^q_{\varphi_2})$ fin-accepts w^1 *(definition of $\mathcal{A}_\varphi$)*

iff $\mathcal{A}_{\varphi_1}$ fin-accepts w or $\mathcal{A}_{\varphi_2}$ fin-accepts w *(Proposition 2.3.15)* iff $w \models \varphi_1$ or $w \models \varphi_2$ *(induction hypothesis)*

iff $w \models (\varphi_1 \vee \varphi_2)$ *(semantics of LTL)*

$(\varphi = (\varphi_1 \wedge \varphi_2))$

$\mathcal{A}_\varphi$ fin-accepts w

iff there exists a transition $\langle q_I, \theta, F, Q' \rangle \in \Delta$ such that $w(0) \models \theta$ and for all $q \in Q'$, $\mathcal{A}_\varphi^q$ fin-accepts w^1 *(Proposition 2.3.15)*

iff there exist transitions $\langle q_{I1}, \theta_1, F_1, Q'_1 \rangle \in \Delta_1$ and $\langle q_{I2}, \theta_2, F_2, Q'_2 \rangle \in \Delta_2$ such that $w(0) \models (\theta_1 \wedge \theta_2)$ and for all $q \in Q'_1 \cup Q'_2$, $\mathcal{A}_\varphi^q$ fin-accepts w^1 *(definition of $\mathcal{A}_\varphi$)*

iff there exists a transition $\langle q_{I1}, \theta_1, F_1, Q'_1 \rangle \in \Delta_1$ such that $w(0) \models \theta_1$ and for all $q \in Q'_1$, $\mathcal{A}_\varphi^q$ $(= \mathcal{A}_{\varphi_1}^q)$ fin-accepts w^1

and

there exists a transition $\langle q_{I2}, \theta_2, F_2, Q'_2 \rangle \in \Delta_2$ such that $w(0) \models \theta_2$ and for all $q \in Q'_2$, $\mathcal{A}_\varphi^q$ $(= \mathcal{A}_{\varphi_2}^q)$ fin-accepts w^1

iff $\mathcal{A}_{\varphi_1}$ fin-accepts w *and* $\mathcal{A}_{\varphi_2}$ fin-accepts w *(Proposition 2.3.15)*

iff $w \models \varphi_1$ *and* $w \models \varphi_2$ *(induction hypothesis)*

iff $w \models (\varphi_1 \wedge \varphi_2)$ *(semantics of LTL)*

$(\varphi = (\varphi_1 \, \mathsf{U_s} \, \varphi_2)$ *or* $\varphi = (\varphi_1 \, \mathsf{U_w} \, \varphi_2))$

$\mathcal{A}_\varphi$ fin-accepts w

iff there exists an index $0 \leq i < \omega$ such that $\mathcal{A}_{\varphi_2}$ fin-accepts w^i and for all $0 \leq j < i$, $\mathcal{A}_{\varphi_1}$ fin-accepts w^j

or

$\varphi = (\varphi_1 \, \mathsf{U_w} \, \varphi_2)$, and for all $0 \leq i < \omega$, $\mathcal{A}_{\varphi_1}$ fin-accepts w^i *(Lemma 3.3.1)*

iff there exists an index $0 \leq i < \omega$ such that $w^i \models \varphi_2$ and for all $0 \leq j < i$, $w^j \models \varphi_1$

or

$\varphi = (\varphi_1 \, \mathsf{U_w} \, \varphi_2)$, and for all $0 \leq i < \omega$, $w^i \models \varphi_1$ *(induction hypothesis)*

iff $w \models \varphi$ *(semantics of LTL)*

$(\varphi = (\varphi_1 \, \mathsf{R_s} \, \varphi_2)$ *or* $\varphi = (\varphi_1 \, \mathsf{R_w} \, \varphi_2))$

$\mathcal{A}_\varphi$ fin-accepts w

iff $\mathcal{A}_{(\varphi_2 \circ (\varphi_1 \wedge \varphi_2))}$ fin-accepts w, where $\circ$ is an Until connective of the same strength as the main connective of φ *(definition of $\mathcal{A}_\varphi$)*

iff there exists an index $0 \leq i < \omega$ such that $\mathcal{A}_{(\varphi_1 \wedge \varphi_2)}$ fin-accepts w^i and for all $0 \leq j < i$, $\mathcal{A}_{\varphi_2}$ fin-accepts w^j

or

$\circ = \mathsf{U_w}$, and $\mathcal{A}_{\varphi_2}$ fin-accepts w^i for all $0 \leq i < \omega$ *(Lemma 3.3.1)*

iff there exists an index $0 \leq i < \omega$ such that $w^i \models (\varphi_1 \wedge \varphi_2)$ and for all $0 \leq j < i$, $w^j \models \varphi_2$

or

$\circ = \mathsf{U_w}$, and $w^i \models \varphi_2$ for all $0 \leq i < \omega$

(case "$\wedge$" and the induction hypothesis)

$$\text{iff} \quad w \models \big(\varphi_2 \circ (\varphi_1 \wedge \varphi_2)\big) \qquad \textit{(semantics of LTL)}$$
$$\text{iff} \quad w \models \varphi \qquad \textit{(semantics of LTL)}$$

The result holds by induction for all formulas $\varphi \in LTL^{\mathrm{PNF}}(AP)$. $\square$

3.4 REVERSE TRANSLATION

In this section we verify that, for any self-loop alternating automaton $\mathcal{A}$ over an alphabet Σ with 2^n elements for some $n \in \mathbb{N}$ (i.e., a set that is equipollent to a powerset of some finite set S with n elements), there exists an LTL formula φ over the atomic propositions S such that for all $w \in \Sigma^\omega$, $\mathcal{A}$ fin-accepts w iff $w \models \varphi$ holds. Together with Theorem 3.3.2, this result establishes the expressive equivalence between self-loop alternating automata and LTL. Proofs for this result (based on slightly different basic definitions and notions of acceptance) have previously been presented by Rohde [1997], and Löding and Thomas [2000]; in this section, we prove the result directly for automata having multiple acceptance conditions on transitions instead of states. Additionally, we consider also the complexity of the reverse translation by finding upper bounds for the number of subformulas and temporal subformulas in the LTL formulas built from self-loop alternating automata via reverse translation.

As noted by Rohde [1997], the expressive equivalence of self-loop alternating automata and LTL generalizes to self-loop alternating automata over an arbitrary (finite) nonempty alphabet Σ in the sense that for any self-loop alternating automaton $\mathcal{A}$ with alphabet Σ, there exists a finite set S, a one-to-one mapping $\alpha : \Sigma \to 2^S$ and an LTL formula $\varphi \in LTL(S)$ such that for all $w \in \Sigma^\omega$, $\mathcal{A}$ fin-accepts w iff $\big(\alpha(w(i))\big)_{0 \le i < \omega} \models \varphi$ holds. It is easy to define a one-to-one mapping α by taking S to be any finite set with at least $\lceil \log_2 |\Sigma| \rceil$ elements. The claim then follows by applying the basic correspondence between LTL and self-loop alternating automata whose alphabet's size is a power of 2 to the self-loop alternating automaton (over the alphabet 2^S) obtained from $\mathcal{A}$ by replacing each element of each transition guard of $\mathcal{A}$ with its image under α.

We begin by characterizing in LTL the behavior of a copy of a (subautomaton of a) self-loop alternating automaton $\mathcal{A} = \langle 2^S, Q, \Delta, q_I, \mathcal{F} \rangle$ along a path in a fin-accepting run of the automaton on some input $w \in (2^S)^\omega$. Interpreting the path as the description of the stepwise behavior of the copy of the automaton, we see (cf. Fig. 2.5, p. 28) that the copy either "stays" in the state $q \in Q$ labeling the first node in the path for a finite number of steps and then exits the state (without ever entering it again, because $\mathcal{A}$ is a self-loop alternating automaton), or remains in the state indefinitely by taking only self-loop transitions (i.e., spawning a new copy of itself at every step). Because the run is fin-accepting, the infinite chain formed from these self-loops corresponds to a fin-accepting branch of the run. Hence, for all acceptance conditions $f \in \mathcal{F}$, the copy of the automaton takes infinitely many self-loops, none of which is an f-transition of $\mathcal{A}$.

To formalize this intuition, we first introduce some notation. Assume that q is the source state of the i^{th} consecutive edge (labeled with a transition $t =$

$\langle q, \Gamma, F, Q' \rangle \in \Delta$) in a chain through the fin-accepting run of $\mathcal{A}$. Because q is the initial state of some subautomaton of $\mathcal{A}$, it follows by Proposition 2.3.15 that the guard of t contains the input symbol $w(i) \in 2^S$ (i.e., $w(i) \models \theta$ holds for the characteristic Boolean formula θ of Γ), and all subautomata rooted at the states in Q' fin-accept w^{i+1}. Suppose that the language accepted by each subautomaton rooted at a target state $q' \in Q' \setminus \{q\}$ of t coincides with the language of some LTL formula $\varphi_{q'}$, i.e., $w^{i+1} \models \varphi_{q'}$ holds. We thus find that

$$w^i \models \mu\big(\langle q, \theta, F, Q' \rangle\big) \stackrel{\mathit{def}}{=} (\theta \wedge \bigwedge_{q' \in Q' \setminus \{q\}} \mathsf{X}\varphi_{q'}).$$

Using μ, the implications of a copy of $\mathcal{A}$ (that is in state $q \in Q$) taking a self-loop, a non-self-loop, or a self-loop that is not an f-transition of $\mathcal{A}$ can now be written as the LTL formulas

$$\psi_{\text{self-loop}}(q) \stackrel{\mathit{def}}{=} \bigvee_{\substack{\langle q, \theta, F, Q' \rangle \in \Delta, \\ q \in Q'}} \mu\big(\langle q, \theta, F, Q' \rangle\big), \qquad \psi_{\text{non-self-loop}}(q) \stackrel{\mathit{def}}{=} \bigvee_{\substack{\langle q, \theta, F, Q' \rangle \in \Delta, \\ q \notin Q'}} \mu\big(\langle q, \theta, F, Q' \rangle\big)$$

and

$$\psi_{\text{avoid}}(q, f) \stackrel{\mathit{def}}{=} \bigvee_{\substack{\langle q, \theta, F, Q' \rangle \in \Delta, \\ q \in Q',\ f \notin F}} \mu\big(\langle q, \theta, F, Q' \rangle\big).$$

We can now give a characterization of the above description of the looping behavior of $\mathcal{A}$ in LTL. Assuming the existence of LTL formulas corresponding to the successors of the state q of the alternating automaton $\mathcal{A}$ (excluding q itself), we can apply the following lemma to find an LTL formula, the language of which coincides with the language fin-accepted by the subautomaton $\mathcal{A}^q$.

Lemma 3.4.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton over the alphabet $\Sigma = 2^S$ for some finite set S, and let $q \in Q$. Assume that for all successors q' of q in $\mathcal{A}$, excluding q itself, there exists an LTL formula $\varphi_{q'}$ such that for all $w \in \Sigma^\omega$, the subautomaton $(\mathcal{A}^q)^{q'}$ fin-accepts w iff $w \models \varphi_{q'}$ holds. For all $w \in \Sigma^\omega$, $\mathcal{A}^q$ fin-accepts w iff w satisfies the formula*

$$\varphi_q \stackrel{\mathit{def}}{=} \Big(\big(\psi_{\text{self-loop}}(q) \,\mathsf{U_s}\, \psi_{\text{non-self-loop}}(q)\big) \vee \mathsf{G}\big(\psi_{\text{self-loop}}(q) \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(q, f)\big)\Big).$$

Proof: (Only if) Let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}^q$ on some $w \in \Sigma^\omega$. By Proposition 2.3.7, there exists an index $0 \leq i \leq \omega$ and a chain of edges $(e_j)_{0 \leq j < i+1}$, $e_j = \langle v_j, V'_j \rangle \in E \cap (V_j \times 2^{V_{j+1}})$, such that $L(e_j)$ is an initial self-loop of $\mathcal{A}^q$ for all $0 \leq j < i$, and if $i < \omega$, then $L(e_i)$ is an initial transition of $\mathcal{A}^q$ that is not a self-loop. It is clear that $L(v_j) = q$ holds for all $0 \leq j < i+1$.

Let $0 \leq j < i+1$. Because G is a run, the edge $e_j = \langle v_j, V'_j \rangle$ is labeled with a transition $t_j = \langle q, \theta_j, F_j, Q'_j \rangle \in \Delta$ such that $w(j) \models \theta_j$ and $Q'_j = L(V'_j)$ hold. Because θ_j is a Boolean formula, it follows that $w^j \models \theta_j$ holds, and because G is fin-accepting, $(\mathcal{A}^q)^{q'}$ fin-accepts w^{j+1} for all $q' \in Q'_j$ by Proposition 2.3.9. By assumption, this implies that $w^{j+1} \models \varphi_{q'}$ holds for all

$q' \in Q'_j \setminus \{q\}$. Therefore $w^j \models \big(\theta_j \wedge \bigwedge_{q' \in Q'_j \setminus \{q\}} \mathsf{X}\varphi_{q'}\big)$, i.e., $w^j \models \mu(t_j)$ holds by the semantics of LTL.

Because $L(e_j)$ is an initial self-loop of $\mathcal{A}^q$ for all $0 \leq j < i$, it follows from the definition of $\psi_{\text{self-loop}}(q)$ that $w^j \models \psi_{\text{self-loop}}(q)$ holds for all $0 \leq j < i$. If $i < \omega$, then, because $L(e_i)$ is an initial transition of $\mathcal{A}^q$ that is not a self-loop, also $w^i \models \psi_{\text{non-self-loop}}(q)$ holds. But then $w \models \big(\psi_{\text{self-loop}}(q) \, \mathsf{U_s} \, \psi_{\text{non-self-loop}}(q)\big)$ holds by the semantics of LTL, which implies that $w \models \varphi_q$.

If $i = \omega$, then the chain $\beta \overset{\text{def}}{=} (e_j)_{0 \leq j < i}$ is an infinite branch, all edges of which are labeled with initial self-loops of $\mathcal{A}^q$. As seen above, $w^j \models \mu(t_j)$ and $w^j \models \psi_{\text{self-loop}}(q)$ hold for all $0 \leq j < \omega$. Fix $0 \leq j < \omega$, and let $f \in \mathcal{F}$. Because G is fin-accepting, $\text{fin}(\beta) = \emptyset$, and thus there exists an index $j \leq k < \omega$ such that $L(e_k)$ is not an f-transition of $\mathcal{A}$. Because $w^k \models \mu(t_k)$ holds, and $L(e_k)$ is an initial self-loop of $\mathcal{A}^q$, it follows that $w^k \models \psi_{\text{avoid}}(q, f)$ holds. But then, because $k \geq j$, $w^j \models \mathsf{F}\psi_{\text{avoid}}(q, f)$ holds, and because f is arbitrary and $w^j \models \psi_{\text{self-loop}}(q)$, it follows that $w^j \models \big(\psi_{\text{self-loop}}(q) \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(q, f)\big)$ holds. Since j is arbitrary, we conclude that $w \models \varphi_q$ holds also in this case.

(If) Assume that $w \models \varphi_q$ holds. Then w satisfies at least one of the formulas $\big(\psi_{\text{self-loop}}(q) \, \mathsf{U_s} \, \psi_{\text{non-self-loop}}(q)\big)$ and $\mathsf{G}\big(\psi_{\text{self-loop}}(q) \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(q, f)\big)$, and there necessarily exists a maximal index $0 \leq i \leq \omega$ such that $w^j \models \psi_{\text{self-loop}}(q)$ and $w^j \not\models \psi_{\text{non-self-loop}}(q)$ hold for all $0 \leq j < i$, and if $i < \omega$, then $w^i \models \psi_{\text{non-self-loop}}(q)$. From the definition of $\psi_{\text{self-loop}}(q)$ it follows that the set of initial self-loop transitions $T_j \overset{\text{def}}{=} \big\{\langle q, \theta, F, Q' \rangle \in \Delta \,\big|\, q \in Q', w^j \models \mu\big(\langle q, \theta, F, Q' \rangle\big)\big\}$ is nonempty for all $0 \leq j < i$. Our goal is to choose self-loops $t_j \overset{\text{def}}{=} \langle q, \theta_j, F_j, Q'_j \rangle \in T_j$ for all $0 \leq j < i$ (and if $i < \omega$, an additional non-self-loop transition $t_i \overset{\text{def}}{=} \langle q, \theta_i, F_i, Q'_i \rangle \in \Delta$ such that $w^i \models \mu(t_i)$ holds; t_i exists because $w^i \models \psi_{\text{non-self-loop}}(q)$ holds in this case) and construct a fin-accepting semi-run G of $\mathcal{A}^q$ on w by forming a (possibly infinite) chain of edges labeled with these transitions. For this purpose, we fix a total order $\prec$ on the set of acceptance conditions $\mathcal{F}$; because $\mathcal{F}$ is finite, every nonempty subset of $\mathcal{F}$ then contains a minimal element under $\prec$.

(Definition of the transitions t_j) Let $t_0 \in T_0$ be any element of T_0. Assume that the transitions $t_j = \langle q, \theta_j, F_j, Q'_j \rangle$ have already been defined for all $0 \leq j < k$ for some $1 \leq k < i$. Let $\alpha_k \overset{\text{def}}{=} \min\big(\{k-1\} \cup \big\{0 \leq \ell < k \,\big|\, \bigcap_{\ell \leq j \leq k-1} F_j \neq \emptyset\big\}\big)$ be the minimal index strictly less than k such that all transitions $t_{\alpha_k}, t_{\alpha_k+1}, \ldots, t_{k-1}$ share a common acceptance condition in $\mathcal{F}$ if such a condition exists (and let $\alpha_k \overset{\text{def}}{=} k-1$ otherwise). Let $\widetilde{F}_k \overset{\text{def}}{=} \bigcap_{\alpha_k \leq j \leq k-1} F_j$ be the set of all acceptance conditions shared by these transitions, and define $\tilde{f}_k \overset{\text{def}}{=} \min_{\prec} \widetilde{F}_k$ for every nonempty $\widetilde{F}_k$. Now, let $t_k \overset{\text{def}}{=} \langle q, \theta_k, F_k, Q'_k \rangle$ be any transition $t = \langle q, \theta, F, Q' \rangle \in T_k$ such that $\tilde{f}_k \notin F$ if $\widetilde{F}_k \neq \emptyset$ and such a transition exists in T_k; otherwise let t_k be any transition in the set T_k.

(Definition of G) Without loss of generality, we may write $Q'_j \setminus \{q\}$ as a finite set of distinct states $Q'_j = \{q_{j,1}, q_{j,2}, \ldots, q_{j,n_j}\}$ for some $0 \leq n_j < \omega$ and all $0 \leq j < i+1$. Let $G = \langle V, E, L \rangle$, where

- $V_0 \overset{\text{def}}{=} \{v_0\}$, $V_{j+1} \overset{\text{def}}{=} \{v_{j+1}, v_{j,1}, \ldots, v_{j,n_j}\}$ for all $0 \leq j < i$, and if $i < \omega$, let $V_{i+1} \overset{\text{def}}{=} \{v_{i,1}, \ldots, v_{i,n_i}\}$ and $V_j \overset{\text{def}}{=} \emptyset$ for all $i+1 < j < \omega$;

- $E \stackrel{def}{=} \bigcup_{0 \le j < i+1} \{\langle v_j, V_{j+1} \rangle\}$;
- $L(v_j) \stackrel{def}{=} q$, $L(v_{j,k}) \stackrel{def}{=} q_{j,k}$, and $L(\langle v_j, V_{j+1} \rangle) \stackrel{def}{=} t_j$ for all $0 \le j < i+1$ and $1 \le k \le n_j$.

(The structure of the graph G is identical to the graph defined in the proof of the "If" direction of Lemma 3.3.1; see also Fig. 3.5.)

(G is a semi-run of $\mathcal{A}^q$ on w) We check that G satisfies all constraints required of a semi-run of $\mathcal{A}^q$ on w.

(Partitioning) $V_0 = \{v_0\}$, and V is partitioned into finite disjoint levels (with edges between successive levels of G) by construction.

(Causality) Let $v \in V_j$ for some $0 \le j < \omega$. Then v either has no outgoing edges, or $v = v_j$ and $j < i+1$. In this case v has the unique outgoing edge $e = \langle v_j, V_{j+1} \rangle \in E$. On the other hand, because the target node set of the only edge starting from a node at level $0 \le j < i+1$ covers all nodes in V_{j+1}, it is clear that each node in V_j for some $1 \le j < \omega$ is a successor of some node at level $j-1$.

(Consistency of L) Obviously, $L(v_0) = q$ holds. Let $e \in E$. By construction, $e = \langle v_j, V_{j+1} \rangle$ holds for some $0 \le j < i+1$. Since $w^j \models \mu(t_j)$ holds (i.e., $w^j \models \big(\theta_j \wedge \bigwedge_{q' \in Q'_j \setminus \{q\}} \mathsf{X}\varphi_{q'}\big)$ holds), $w(j) \models \theta_j$ holds, and because $L(e) = t_j = \langle q, \theta_j, F_j, Q'_j \rangle = \big\langle L(v_j), \theta_j, F_j, L(V_{j+1}) \big\rangle$ holds, the labeling L is consistent.

(G is fin-accepting) We check that G is fin-accepting. If $i < \omega$ (i.e., if $w \models (\psi_{\text{self-loop}}(q) \,\mathsf{U_s}\, \psi_{\text{non-self-loop}}(q))$ holds), then the edge set E is finite. This implies that $\mathcal{B}(G) = \emptyset$, and thus G is trivially a fin-accepting semi-run of $\mathcal{A}^q$ on w. Otherwise $\mathcal{B}(G)$ contains a unique infinite branch $\beta \stackrel{def}{=} (e_j)_{0 \le j < \omega}$, where $e_j = \langle v_j, V_{j+1} \rangle$ for all $0 \le j < \omega$.

Assume that β is not fin-accepting. Therefore, there exists a minimal index $0 \le k < \omega$ such that all transitions $L(e_j) = t_j = \langle q, \theta_j, F_j, Q'_j \rangle$ share a nonempty set of acceptance conditions from $\mathcal{F}$ for all $k \le j < \omega$. Let $f_{\min}$ be the $\prec$-minimal acceptance condition among the maximal set of such conditions. Because k is minimal, there exists another index $k \le k' < \omega$ such that $f_{\min}$ is also the minimal element of $\widetilde{F}_j$ for all $k' < j < \omega$.

Because $i = \omega$, $w \not\models \big(\psi_{\text{self-loop}}(q) \,\mathsf{U_s}\, \psi_{\text{non-self-loop}}(q)\big)$. Therefore it is necessarily the case that $w \models \mathsf{G}\big(\psi_{\text{self-loop}}(q) \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(q, f)\big)$ holds, and thus there exists an index $k' < \ell < \omega$ such that $w^\ell \models \psi_{\text{avoid}}(q, f_{\min}) = \bigvee_{\substack{\langle q,\theta,F,Q' \rangle \in \Delta \\ q \in Q',\, f_{\min} \notin F}} \mu\big(\langle q, \theta, F, Q' \rangle\big)$. It now follows that T_ℓ has a nonempty subset T of transitions, none of which includes $f_{\min}$ in its acceptance conditions. Because $f_{\min}$ is the minimal element of $\widetilde{F}_\ell$, it follows that t_ℓ was chosen from the set T. But then $f_{\min}$ cannot be one of the acceptance conditions shared by all transitions t_j for all $k \le j < \omega$, which is a contradiction. It follows that β is fin-accepting, and G is a fin-accepting semi-run of $\mathcal{A}^q$ on w.

(G can be extended into a fin-accepting run of $\mathcal{A}^q$ on w) Let $v \in V$ be a node with no outgoing edges in G. Then $v = v_{j,k} \in V_{j+1}$ holds for some $0 \le j < i+1$ and $1 \le k \le n_j$ such that $L(v) = q_{j,k} \in Q'_j \setminus \{q\}$. Because $w^j \models \mu(t_j)$ holds, that is, $w^j \models \big(\theta_j \wedge \bigwedge_{q' \in Q'_j \setminus \{q\}} \mathsf{X}\varphi_{q'}\big)$ holds, it follows that

$w^{j+1} \models \varphi_{q'}$ for all $q' \in Q'_j \setminus \{q\}$. In particular, $(\mathcal{A}^q)^{q_{j,k}}$ now has a fin-accepting run on w^{j+1} by assumption. Since v is an arbitrary node of G with no outgoing edges, it follows that G can be extended to a full fin-accepting run of $\mathcal{A}^q$ on w by Proposition 2.3.14, and thus $\mathcal{A}^q$ fin-accepts w. □

We can now establish the main result of this section by a straightforward inductive proof on the structure of self-loop alternating automata, using Lemma 3.4.1 for proving the induction step.

Theorem 3.4.2 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton over the alphabet $\Sigma = 2^S$ for some finite set S. There exists an LTL formula φ over the atomic propositions S such that for all $w \in \Sigma^\omega$, $\mathcal{A}$ fin-accepts w iff $w \models \varphi$ holds.*

Proof: Because $\mathcal{A}$ is a self-loop alternating automaton, there exists a function $\rho : Q \to \mathbb{N}$ such that for all transitions $\langle q, \Gamma, F, Q' \rangle \in \Delta$, $\rho(q') < \rho(q)$ holds for all $q' \in Q' \setminus \{q\}$. In particular, as seen in the proof of Proposition 2.3.16, $\rho(q)$ can be defined as

$$\rho(q) \stackrel{def}{=} \max \big\{ |x| \bigm| x \text{ is a simple path from } q \text{ to a state } q' \in Q_0 \big\}$$

where $Q_0 \stackrel{def}{=} \big\{ q \in Q \bigm| \text{for all } \langle q, \Gamma, F, Q' \rangle \in \Delta,\ Q' \subseteq \{q\} \big\}$. We proceed by induction on $\rho(q)$. If $\rho(q) = 1$, then the result follows immediately for the subautomaton $\mathcal{A}^q$ by Lemma 3.4.1, since q has no successors different from itself (and thus the assumption needed in Lemma 3.4.1 holds trivially). Assume that the result holds for all subautomata $\mathcal{A}^q$, where $q \in Q$ satisfies $\rho(q) \le i$ for some $1 \le i < \omega$. Let $q \in Q$ be a state for which $\rho(q) = i + 1$. Then, $\rho(q') \le i$ holds for all successors of q excluding q itself, and the result follows again for the subautomaton $\mathcal{A}^q$ by Lemma 3.4.1 and the induction hypothesis. By induction, we conclude that the result holds also for the subautomaton $\mathcal{A}^{q_I}$, because $\rho(q_I)$ is finite, and therefore also for the automaton $\mathcal{A}$, because $\mathcal{A}$ and $\mathcal{A}^{q_I}$ are fin-equivalent (Proposition 2.3.12). □

The proof of Theorem 3.4.2 gives an inductive procedure for finding an LTL formula that corresponds to a given self-loop alternating automaton by repeatedly using the formula given in Lemma 3.4.1 as a pattern for defining LTL formulas corresponding to states with increasing values of ρ. We illustrate the reverse translation with the following example.

Example 3.4.3 Consider again the self-loop alternating automaton working on the alphabet $\Sigma = \{a, b, c\}$ from Ex. 2.3.21 (repeated in Fig. 3.6 with the values of the function ρ used in Theorem 3.4.2). Because the alphabet Σ consists of three distinct symbols, the automaton can be translated into an LTL formula over two atomic propositions $AP \stackrel{def}{=} \{p_1, p_2\}$ by defining (as described in the beginning of this section) a mapping $\alpha : \Sigma \to 2^{AP}$ with (for example) $\alpha(a) \stackrel{def}{=} \{p_1\}$, $\alpha(b) \stackrel{def}{=} \{p_2\}$ and $\alpha(c) \stackrel{def}{=} \{p_1, p_2\}$. Under this mapping, the guards of transitions in the automaton can be represented with

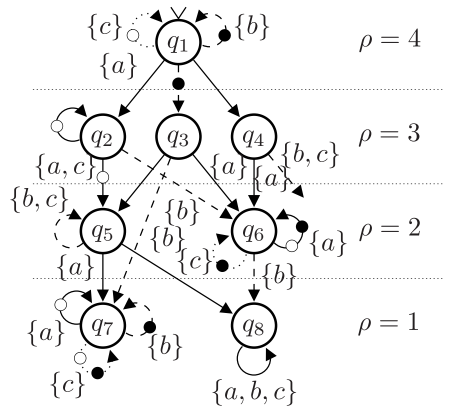


Fig. 3.6: The self-loop alternating automaton of Fig. 2.8 with the values of the function ρ displayed

the characteristic Boolean formulas

$$\begin{aligned}
\theta_{\{a\}} &\stackrel{def}{=} (p_1 \wedge \neg p_2),\\
\theta_{\{b\}} &\stackrel{def}{=} (\neg p_1 \wedge p_2),\\
\theta_{\{c\}} &\stackrel{def}{=} (p_1 \wedge p_2),\\
\theta_{\{a,c\}} &\stackrel{def}{=} \big((p_1 \wedge \neg p_2) \vee (p_1 \wedge p_2)\big) \equiv p_1,\\
\theta_{\{b,c\}} &\stackrel{def}{=} \big((\neg p_1 \wedge p_2) \vee (p_1 \wedge p_2)\big) \equiv p_2, \text{ and}\\
\theta_{\{a,b,c\}} &\stackrel{def}{=} \big((p_1 \wedge \neg p_2) \vee \big((\neg p_1 \wedge p_2) \vee (p_1 \wedge p_2)\big)\big) \equiv (p_1 \vee p_2)
\end{aligned}$$

In the following, we shall use the above θ formulas instead of writing the characteristic Boolean formulas explicitly to simplify the notation. We shall also omit parentheses from formulas whenever it is possible to do so without semantic ambiguity.

The reverse translation can be started from any state q with $\rho(q) = 1$, for example, the state q_7. For this state, we define

$$\begin{aligned}
\psi_{\text{self-loop}}(q_7) &\stackrel{def}{=} \bigvee\nolimits_{\substack{\langle q_7,\theta,F,Q'\rangle\in\Delta,\\ q_7\in Q'}} \mu\big(\langle q_7,\theta,F,Q'\rangle\big)\\
&= \mu\big(\langle q_7,\theta_{\{a\}},\{\circ\},\{q_7\}\rangle\big) \vee \mu\big(\langle q_7,\theta_{\{b\}},\{\bullet\},\{q_7\}\rangle\big)\\
&\qquad \vee \mu\big(\langle q_7,\theta_{\{c\}},\{\bullet,\circ\},\{q_7\}\rangle\big)\\
&= \big(\theta_{\{a\}} \wedge \textstyle\bigwedge_{q'\in\emptyset} \mathsf{X}\varphi_{q'}\big) \vee \big(\theta_{\{b\}} \wedge \textstyle\bigwedge_{q'\in\emptyset} \mathsf{X}\varphi_{q'}\big)\\
&\qquad \vee \big(\theta_{\{c\}} \wedge \textstyle\bigwedge_{q'\in\emptyset} \mathsf{X}\varphi_{q'}\big)\\
&\equiv (\theta_{\{a\}} \wedge \top) \vee (\theta_{\{b\}} \wedge \top) \vee (\theta_{\{c\}} \wedge \top)\\
&\equiv \theta_{\{a\}} \vee \theta_{\{b\}} \vee \theta_{\{c\}},
\end{aligned}$$

$$\psi_{\text{non-self-loop}}(q_7) \stackrel{def}{=} \bigvee\nolimits_{\substack{\langle q_7,\theta,F,Q'\rangle\in\Delta,\\ q_7\notin Q'}} \mu\big(\langle q_7,\theta,F,Q'\rangle\big) \equiv \bot,$$

$$\begin{aligned}
\psi_{\text{avoid}}(q_7,\bullet) &\stackrel{def}{=} \bigvee\nolimits_{\substack{\langle q_7,\theta,F,Q'\rangle\in\Delta,\\ q_7\in Q',\ \bullet\notin F}} \mu\big(\langle q_7,\theta,F,Q'\rangle\big)\\
&= \mu\big(\langle q_7,\theta_{\{a\}},\{\circ\},\{q_7\}\rangle\big)\\
&= \theta_{\{a\}} \wedge \textstyle\bigwedge_{q'\in\emptyset} \mathsf{X}\varphi_{q'}\\
&\equiv \theta_{\{a\}}, \text{ and}
\end{aligned}$$

Table 3.5: $\psi_{\text{self-loop}}$-, $\psi_{\text{non-self-loop}}$- and ψ_{avoid}-formulas built during reverse translation

State q	$\psi_{\text{self-loop}}(q)$	$\psi_{\text{non-self-loop}}(q)$	$\psi_{\text{avoid}}(q,\bullet)$	$\psi_{\text{avoid}}(q,\circ)$
q_7	$\theta_{\{a\}} \vee \theta_{\{b\}} \vee \theta_{\{c\}}$	$\perp$	$\theta_{\{a\}}$	$\theta_{\{b\}}$
q_8	$\theta_{\{a,b,c\}}$	$\perp$	$\theta_{\{a,b,c\}}$	$\theta_{\{a,b,c\}}$
q_5	$\theta_{\{b,c\}}$	$\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_7} \wedge \mathsf{X}\varphi_{q_8}$	$\theta_{\{b,c\}}$	$\theta_{\{b,c\}}$
q_6	$\theta_{\{a\}} \vee \theta_{\{c\}}$	$\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_8}$	$\perp$	$\theta_{\{c\}}$
q_2	$\theta_{\{a,c\}} \wedge \mathsf{X}\varphi_{q_5}$	$\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_6}$	$\theta_{\{a,c\}} \wedge \mathsf{X}\varphi_{q_5}$	$\perp$
q_3	$\perp$	$(\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_5} \wedge \mathsf{X}\varphi_{q_6}) \vee (\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_7})$	$\perp$	$\perp$
q_4	$\perp$	$(\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_6}) \vee \theta_{\{b,c\}}$	$\perp$	$\perp$
q_1	$(\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_3}) \vee \theta_{\{c\}}$	$\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_2} \wedge \mathsf{X}\varphi_{q_4}$	$\theta_{\{c\}}$	$\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_3}$

$$
\begin{aligned}
\psi_{\text{avoid}}(q_7,\circ) &\stackrel{\mathit{def}}{=} \bigvee\nolimits_{\substack{\langle q_7,\theta,F,Q'\rangle\in\Delta,\\ q_7\in Q',\ \circ\notin F}} \mu\big(\langle q_7,\theta,F,Q'\rangle\big)\\
&= \mu\big(\langle q_7,\theta_{\{b\}},\{\bullet\},\{q_7\}\rangle\big)\\
&= \theta_{\{b\}} \wedge \bigwedge\nolimits_{q'\in\emptyset} \mathsf{X}\varphi_{q'}\\
&\equiv \theta_{\{b\}}.
\end{aligned}
$$

Applying Lemma 3.4.1, we get the formula

$$
\begin{aligned}
\varphi_{q_7} &\stackrel{\mathit{def}}{=} \big(\psi_{\text{self-loop}}(q_7)\,\mathsf{U_s}\,\psi_{\text{non-self-loop}}(q_7)\big)\\
&\qquad\qquad\qquad \vee \mathsf{G}\big(\psi_{\text{self-loop}}(q_7) \wedge \bigwedge\nolimits_{f\in\{\bullet,\circ\}} \mathsf{F}\psi_{\text{avoid}}(q_7,f)\big)\\
&\equiv \big((\theta_{\{a\}} \vee \theta_{\{b\}} \vee \theta_{\{c\}})\,\mathsf{U_s}\perp\big) \vee \mathsf{G}\big((\theta_{\{a\}} \vee \theta_{\{b\}} \vee \theta_{\{c\}}) \wedge \mathsf{F}\theta_{\{a\}} \wedge \mathsf{F}\theta_{\{b\}}\big)\\
&\equiv \perp \vee\, \mathsf{G}\big((\theta_{\{a\}} \vee \theta_{\{b\}} \vee \theta_{\{c\}}) \wedge \mathsf{F}\theta_{\{a\}} \wedge \mathsf{F}\theta_{\{b\}}\big)\\
&\equiv \mathsf{G}\big((\theta_{\{a\}} \vee \theta_{\{b\}} \vee \theta_{\{c\}}) \wedge \mathsf{F}\theta_{\{a\}} \wedge \mathsf{F}\theta_{\{b\}}\big).
\end{aligned}
$$

Table 3.5 and Table 3.6 show (simplified) formulas built by repeating the reverse translation in the states q_8, q_5, q_6, q_2, q_3, q_4 and q_1 (where the order is determined by increasing values of ρ). No (simplified) formula corresponding to a transient state of the automaton has a top-level subformula having G as its main connective. By the discussion at the beginning of this section and Theorem 3.4.2, the automaton fin-accepts a word $w \in \{a,b,c\}^\omega$ iff $\big(\alpha(w(i))\big)_{0\le i<\omega} \models \varphi_{q_1}$ holds. ■

It is easy to see from the previous example that the length of the formulas obtained by repeated applications of Lemma 3.4.1 grows rapidly as the value of ρ increases (in the example, we did not even try to write the φ_q-formulas in explicit form for states with $\rho(q) > 1$). There is nevertheless some sharing between the subformulas: in the remainder of this sec-

Table 3.6: φ_q-formulas built during reverse translation

State q	φ_q
q_7	$\mathsf{G}\big((\theta_{\{a\}} \vee \theta_{\{b\}} \vee \theta_{\{c\}}) \wedge \mathsf{F}\theta_{\{a\}} \wedge \mathsf{F}\theta_{\{b\}}\big)$
q_8	$\mathsf{G}\theta_{\{a,b,c\}}$
q_5	$\big(\theta_{\{b,c\}}\,\mathsf{U_s}\,(\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_7} \wedge \mathsf{X}\varphi_{q_8})\big) \vee \mathsf{G}\theta_{\{b,c\}}$
q_6	$(\theta_{\{a\}} \vee \theta_{\{c\}})\,\mathsf{U_s}\,(\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_8})$
q_2	$(\theta_{\{a,c\}} \wedge \mathsf{X}\varphi_{q_5})\,\mathsf{U_s}\,(\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_6})$
q_3	$(\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_5} \wedge \mathsf{X}\varphi_{q_6}) \vee (\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_7})$
q_4	$(\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_6}) \vee \theta_{\{b,c\}}$
q_1	$\big(((\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_3}) \vee \theta_{\{c\}})\,\mathsf{U_s}\,(\theta_{\{a\}} \wedge \mathsf{X}\varphi_{q_2} \wedge \mathsf{X}\varphi_{q_4})\big)$ $\vee\, \mathsf{G}\big(((\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_3}) \vee \theta_{\{c\}}) \wedge \mathsf{F}\theta_{\{c\}} \wedge \mathsf{F}(\theta_{\{b\}} \wedge \mathsf{X}\varphi_{q_3})\big)$

tion, we shall investigate the behavior of the simple reverse translation procedure based on Lemma 3.4.1 by determining upper bounds for the number of subformulas and pure temporal subformulas in a formula φ obtained by applying the procedure to a self-loop alternating automaton $\mathcal{A}$. In particular, the upper bound for the number of pure temporal subformulas can be used as a coarse measure for the efficiency of the reverse translation: because the formula φ could be translated back into an automaton having at most $1 + |\mathrm{Temp}(\varphi)|$ states (Corollary 3.2.2), $|\mathrm{Temp}(\varphi)|$ should ideally not exceed the number of states $|Q|$ in $\mathcal{A}$. However, because the formula pattern defined in Lemma 3.4.1 includes explicit quantification over the acceptance conditions $\mathcal{F}$ of the automaton $\mathcal{A}$, the upper bound for the number of pure temporal subformulas will actually depend on the product of $|Q|$ and $|\mathcal{F}|$. Even worse, the upper bound for the number of subformulas in φ depends explicitly also on the number of transitions in $\mathcal{A}$ (which may be exponential in $|Q|$), and therefore reverse translation using the simple strategy based on the repeated application of Lemma 3.4.1 is unlikely to be feasible in practice except for very small problem instances.

Proposition 3.4.4 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton with $|Q| \neq \emptyset$ and $|\Delta| \neq \emptyset$ over the alphabet $\Sigma = 2^S$ for some finite set S, and let $\varphi \overset{def}{=} \varphi_{q_I}$ be the LTL formula obtained from $\mathcal{A}$ by repeated applications of Lemma 3.4.1 to states in $\mathcal{A}$ as described in Theorem 3.4.2. Let $c \overset{def}{=} \max\big\{|\mathrm{Sub}(\theta)| : \langle q, \theta, F, Q' \rangle \in \Delta\big\} \geq 1$ be the maximum number of (syntactically distinct) subformulas in any Boolean formula occurring as the guard of a transition in Δ. Then,*

(a) $|\mathrm{Sub}(\varphi)| \in O\big((c + |\mathcal{F}|) \cdot |Q| \cdot |\Delta|\big)$, *and*

(b) $|\mathrm{Temp}(\varphi)| \in O\big((1 + |\mathcal{F}|) \cdot |Q|\big)$.

Proof: (a) Clearly, $|\mathrm{Sub}(\varphi)|$ cannot exceed the number of subformulas created in repeated applications of Lemma 3.4.1. We find an upper bound for the number of these formulas. It follows directly from Theorem 3.4.2 and the definitions of μ, φ_q and the various ψ formulas that a subformula φ' built in reverse translation will have one of the following forms:

(1) $\varphi' = \varphi_q$ for some $q \in Q$; *(Theorem 3.4.2)*
(2) $\varphi' = \mathsf{X}\varphi_q$ for some $q \in Q$; *(definition of μ)*
(3) $\varphi' \in \mathrm{Sub}\big(\bigwedge_{q' \in Q' \setminus \{q\}} \mathsf{X}\varphi_{q'}\big) \setminus \bigcup_{q' \in Q} \mathrm{Sub}(\mathsf{X}\varphi_{q'})$ for some transition $\langle q, \theta, F, Q' \rangle \in \Delta$; *(—)*
(4) $\varphi' \in \mathrm{Sub}(\theta)$ for some transition $\langle q, \theta, F, Q' \rangle \in \Delta$ *(—)*
(5) $\varphi' = \mu(t)$ for some transition $t \in \Delta$ *(definitions of the ψ formulas)*
(6) $\varphi' \in \mathrm{Sub}\big(\psi_{\text{self-loop}}(q)\big) \setminus \bigcup_{t \in \Delta} \mathrm{Sub}\big(\mu(t)\big)$ for some $q \in Q$; *(def. of φ_q)*
(7) $\varphi' \in \mathrm{Sub}\big(\psi_{\text{non-self-loop}}(q)\big) \setminus \bigcup_{t \in \Delta} \mathrm{Sub}\big(\mu(t)\big)$ for some $q \in Q$; *(—)*
(8) $\varphi' \in \mathrm{Sub}\big(\psi_{\text{avoid}}(q, f)\big) \setminus \bigcup_{t \in \Delta} \mathrm{Sub}\big(\mu(t)\big)$ for some $q \in Q$ and $f \in \mathcal{F}$; *(—)*
(9) $\varphi' = \mathsf{F}\psi_{\text{avoid}}(q, f)$ for some $q \in Q$ and $f \in \mathcal{F}$; *(—)*
(10) $\varphi' \in \mathrm{Sub}\big(\bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(q, f)\big) \setminus \bigcup_{q' \in Q} \bigcup_{f' \in \mathcal{F}} \mathrm{Sub}\big(\mathsf{F}\psi_{\text{avoid}}(q', f')\big)$ for some $q \in Q$; *(—)*
(11) $\varphi' = \big(\psi_{\text{self-loop}}(q) \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(q, f)\big)$ for some $q \in Q$; *(—)*

(12) $\varphi' = \mathsf{G}\big(\psi_{\text{self-loop}}(q) \wedge \bigwedge_{f\in\mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(q,f)\big)$ for some $q \in Q$; *(—)*

(13) or $\varphi' = \big(\psi_{\text{self-loop}}(q) \,\mathsf{U_s}\, \psi_{\text{non-self-loop}}(q)\big)$ for some $q \in Q$. *(—)*

The total number of subformulas of type (1), (2), (11), (12) and (13) is clearly less than or equal to $5 \cdot |Q|$. Similarly, there are at most $|\Delta|$ subformulas of type (5) and at most $c \cdot |\Delta|$ subformulas of type (4) (each guard of a transition in $\mathcal{A}$ has at most c syntactically distinct subformulas). The number of subformulas of type (9) is obviously bounded by $|Q| \cdot |\mathcal{F}|$.

It is straightforward to check that any formula of the form $\bigwedge_{\varphi\in\Phi}\varphi$ or $\bigvee_{\varphi\in\Phi}\varphi$ for a finite $\Phi \subseteq LTL(AP)$ has no more than $|\Phi|$ subformulas not appearing as a subformula of any $\varphi \in \Phi$, independently of the way in which the conjunction or the disjunction is parenthesized. Therefore, the number of subformulas of type (3) or (10) cannot exceed $|Q| \cdot |\Delta|$ and $|Q| \cdot |\mathcal{F}|$, respectively.

Finally, it is easy to see that, for all $q \in Q$ and $f \in \mathcal{F}$, $\psi_{\text{self-loop}}(q)$, $\psi_{\text{non-self-loop}}(q)$ or $\psi_{\text{avoid}}(q,f)$ is of the form $\bigvee_{t\in\Delta'}\mu(t)$ for some $\Delta' \subseteq \Delta_q \stackrel{def}{=} \big(\{q\} \times 2^{\Sigma} \times 2^{\mathcal{F}} \times 2^{Q}\big) \cap \Delta$. Clearly, $\big|\mathrm{Sub}\big(\bigvee_{t\in\Delta'}\mu(t)\big) \setminus \bigcup_{t\in\Delta}\mathrm{Sub}\big(\mu(t)\big)\big| \leq \big|\mathrm{Sub}\big(\bigvee_{t\in\Delta_q}\mu(t)\big)\big| \leq |\Delta_q|$ holds (for any $q \in Q$) by the above discussion. It follows that at most $\sum_{q\in Q}|\Delta_q| = |\Delta|$ subformulas of type (6) or (7) and $\sum_{q\in Q}|\Delta_q| \cdot |\mathcal{F}| = |\Delta| \cdot |\mathcal{F}|$ subformulas of type (8) are created in reverse translation.

Adding the numbers of the subformulas, we find that $|\mathrm{Sub}(\varphi_{q_I})| \leq \big(5 + 2 \cdot |\mathcal{F}|\big) \cdot |Q| + (3 + c + |\mathcal{F}|) \cdot |\Delta| + |Q| \cdot |\Delta| \in O\big((c + |\mathcal{F}|) \cdot |Q| \cdot |\Delta|\big)$.

(b) By the classification of subformulas of φ in (a), all pure temporal subformulas built in reverse translation are of the form (2), (9), (12) or (13). Therefore, $|\mathrm{Temp}(\varphi)| \leq \big(3 + |\mathcal{F}|\big) \cdot |Q| \in O\big((1 + |\mathcal{F}|) \cdot |Q|\big)$. $\square$

4 NONDETERMINIZATION OF SELF-LOOP ALTERNATING AUTOMATA

In this chapter we study the translation of self-loop alternating automata into nondeterministic automata. Because nondeterministic automata are equally expressive to alternating automata on both finite [Kozen 1976; Chandra and Stockmeyer 1976; Brzozowski and Leiss 1980; Chandra et al. 1981] and infinite inputs (under many notions of acceptance, using the same mode of acceptance for both types of automata) [Miyano and Hayashi 1984a,b; Lindsay 1988; Muller and Schupp 1995], checking whether an alternating automaton recognizes the empty language can be reduced to the corresponding question on a nondeterministic automaton that is equivalent to the alternating one. In the worst case, however, the number of states in the smallest such nondeterministic automaton is exponential in the number of states in the alternating automaton. Nevertheless, the language emptiness problem of alternating automata is usually solved in practice via some form of (explicit or implicit) nondeterminization.

In general, the key to translating an alternating automaton into a finite nondeterministic one is to find a finite encoding for the information that is needed to distinguish accepting branches from non-accepting ones in a run of the automaton. It is well-known that this can be done under very general notions of acceptance by merging branches together in a systematic way to keep the number of active copies at each level of the run of the automaton bounded. We begin this chapter by reviewing this process of *run uniformization* for self-loop alternating automata working in fin-acceptance mode in Sect. 4.1. The uniformization of runs leads to a construction for translating self-loop alternating automata working in fin-acceptance mode into nondeterministic automata working in the same mode (Sect. 4.2). This construction is very similar to the one previously proposed by Gastin and Oddoux [2001]. Because of our more general notion of acceptance, however, a translation procedure (from LTL into nondeterministic automata) based on the construction has inferior performance to Gastin and Oddoux's construction due to its greater worst-case impact on the number of new acceptance conditions required for the nondeterministic automaton. We show in Sect. 4.3 that there is nevertheless no need for the introduction of new acceptance conditions during nondeterminization for a special class of automata which have accepting runs that satisfy certain restrictions on the occurrence of transitions associated with acceptance conditions of the automaton. It occurs that all automata obtained from the translation presented in Ch. 3 trivially belong to this class of automata.

Of course, there are also cases in which an alternating automaton can be translated into a nondeterministic one without an exponential blow-up in the number of states. In Sect. 4.6, we review a simple syntactic subclass of LTL that translates directly into nondeterministic automata using the rules presented in Sect. 3.1. Consequently, any formula in this subclass can be translated into a nondeterministic automaton with a linear number of states in the number of pure temporal subformulas in the formula. This subclass of LTL was previously considered by Schneider [1999] in the context of symbolic translation algorithms between LTL and nondeterministic automata. We show that the subclass is very closely related also to the syntactic subclass LTL^{det} introduced by Maidl [2000a]. We also observe that deciding formula satisfiability in this restricted subset of LTL is **NP**-complete.

4.1 UNIFORM RUNS

In this section, we review a classic technical result that will allow us to restrict the search for an accepting run for a self-loop alternating automaton into a restricted subset of runs in which the number of nodes in each level of a run remains bounded by the number of states in the automaton. This result then leads to a construction for translating self-loop alternating automata into nondeterministic automata (to be discussed in Sect. 4.2).

Merging Run Branches

The only restriction on the size of levels in a run of an alternating automaton is that each level of the run should be finite. The number of nodes in a level of a run may nevertheless grow without any finite bound as the automaton keeps spawning new copies of its subautomata when working on its input. Because the automaton has only finitely many states, however, any level with more nodes than there are states in the automaton represents a situation in which some active copies of the automaton are in the same state. If the run is accepting, then all subgraphs rooted at identically labeled nodes at the level are accepting runs of the same subautomaton on the remaining input (cf. Proposition 2.3.9). A simple idea to reduce the size of the level is to force all copies in the same state to behave identically on the rest of the input by choosing a representative node among the identically labeled nodes and redirecting each edge that covers any of the nonrepresentative nodes as a target node to cover the representative node instead, cutting off the subgraphs originally rooted at the other nodes. Obviously, this transformation causes some branches in the original run to be merged. Repeating the same consideration for each collection of identically labeled nodes at the level would then allow reducing the size of the level to at most as many nodes as there are states in the automaton.

In general, we call a systematic method for merging branches to keep a level of a run finitely bounded a *uniformization strategy*. When applied to the possibly infinite number of levels in the run, however, a uniformization strategy should also preserve the acceptance and non-acceptance of runs in order be useful in considerations on the existence of accepting runs for the automaton. Instead of simply merging branches at their identically labeled nodes, the decision on whether the merging of branches is permissible may in the general case need to be based on additional information (that is, a *memory*) about the past evolution of the individual branches. Despite the possibly unbounded growth in the number of levels that precede another level in a run, the information needed for uniformization can be shown to be representable under very general notions of acceptance using only a finite amount of memory by appealing to the "forgetful determinacy" results of Gurevich and Harrington [1982] (see [Lindsay 1988; Emerson and Jutla

1991; Muller and Schupp 1995]). For example, in the case of automata with a single inf-acceptance condition, this information can be thought of as being annotated directly in the label of each node in the run to make branches which have different memories distinguishable (see, for example, [Isli 1994, 1996]).[1] Ultimately, the finiteness of the memory needed for uniformization will then allow the alternating automaton to be translated into a nondeterministic one which "implements" the uniformization strategy.

As will be shown below, however, no branches ending in the same state need ever be distinguished in runs of self-loop alternating automata with inf- or fin-acceptance, since—similarly to weak [Muller et al. 1986, 1992] or, indeed, very weak [Rohde 1997] automata—the acceptance of an infinite branch in a run of such an automaton is determined by the branch's convergence properties (which, intuitively, do not depend on the past). Therefore, we can use an explicit run-based definition of uniformity. (Instead of reasoning directly about runs of automata, uniformization results are usually presented in a game-theoretic setting by arguing about the existence of finite memory *winning strategies* for infinite acceptance games played on alternating automata. We refer the reader to the article by Muller and Schupp [1995] for a general methodology for constructing such strategies under a variety of notions of acceptance.)

Uniform Runs for Self-loop Automata

Formally, we call a run $G = \langle V, E, L \rangle$ (where V is partitioned into finite disjoint levels $V = \bigcup_{0 \leq i < \omega} V_i$ as usual) of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ *uniform* iff, for all $0 \leq i < \omega$, $L(v) \neq L(v')$ holds for all $v, v' \in V_i$, $v \neq v'$. (Because L is consistent, then obviously $L(v), L(v') \in Q$ holds, and it follows that the size of each level of G is bounded, i.e., $|V_i| \leq |Q|$ holds.) We say that an alternating automaton $\mathcal{A}$ has *uniform inf- (resp. fin-)accepting runs* iff it has a uniform inf- (fin-)accepting run on all words in its language.

Intuitively, an alternating automaton with uniform inf- (fin-)accepting runs is able to accept all words in its language without spawning more than one copy of any of its subautomata at any step when working on an input belonging to the language. Of course, the exact set of subautomata to spawn at a particular step is not necessarily unique due to the possibility of combining initial transitions of the currently active subautomata in several consistent ways, and some of the combinations may fail to give rise to an accepting run for the automaton. Nevertheless, the automaton always has at least one "successful" way to choose the subautomata to spawn at each step whenever the input given for the automaton belongs to the language of the automaton. On the other hand, no such way exists if the input does not belong to this language. This intuition, which is easily seen to apply also to nondeterministic automata and, in fact, all alternating automata with no acceptance conditions, is made formal below.

Proposition 4.1.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a nondeterministic automaton, a self-loop alternating automaton, or an alternating automaton with*

[1] In our graph-based definition of runs, this may necessitate duplicating some nodes in the run to ensure that every node will be uniquely labeled.

$\mathcal{F} = \emptyset$. *For all* $w \in \Sigma^\omega$, $\mathcal{A}$ *fin-accepts* w *iff* $\mathcal{A}$ *has a uniform fin-accepting run on* w.

Proof: (Only if) Let $G = \langle V, E, L\rangle$ be a fin-accepting run of $\mathcal{A}$ on w. If $\mathcal{A}$ is nondeterministic, then, because every transition of $\mathcal{A}$ has exactly one target state, all branches in G are infinite due to forward causality and the consistency of L. Let $\big(\langle v_i, V_i''\rangle\big)_{0\leq i<\omega} \in \mathcal{B}(G)$ be an infinite branch in G (where $\langle v_i, V_i''\rangle \in E \cap (V_i \times 2^{V_{i+1}})$ and $v_{i+1} \in V_i''$ hold for all $0 \leq i < \omega$). It is easy to check that the graph $G' = \langle V', E', L'\rangle$, where $V_i' \overset{def}{=} \{v_i\}$, $L'(v_i) \overset{def}{=} L(v_i)$ $(0 \leq i < \omega)$, $E' \overset{def}{=} \bigcup_{0\leq i<\omega} \big\{\langle v_i, V_{i+1}'\rangle\big\}$ and $L'\big(\langle v_i, V_{i+1}'\rangle\big) \overset{def}{=} L\big(\langle v_i, V_i''\rangle\big)$ $(0 \leq i < \omega)$ is a uniform run of $\mathcal{A}$ on w (to show that L' is consistent, observe that all nodes in V_i'' have the same label in G for all $0 \leq i < \omega$, because $\mathcal{A}$ is nondeterministic). Obviously, G' contains the unique infinite branch $\big(\langle v_i, V_{i+1}'\rangle\big)_{0\leq i<\omega}$; because $\text{fin}\big((\langle v_i, V_{i+1}'\rangle)_{0\leq i<\omega}\big) = \text{fin}\big((\langle v_i, V_i''\rangle)_{0\leq i<\omega}\big) = \emptyset$ holds, G' is fin-accepting.

If $\mathcal{A}$ is an alternating automaton, we apply a uniformization strategy to G to define a uniform fin-accepting run $G' = \langle V', E', L'\rangle$ of $\mathcal{A}$ on w. Intuitively, we choose for each level of G a set of transitions (labeling a subset of edges starting from the level) and define the next level of G' as a set of representative nodes for each distinct state of $\mathcal{A}$ that is a target state of one of these transitions. These representatives then guide the selection of another set of transitions.

Clearly, forming each level of G' from nodes labeled with distinct states of $\mathcal{A}$ already guarantees that the condition that characterizes uniform runs is satisfied. However, the requirement that G' should be a fin-accepting run of $\mathcal{A}$ restricts the choice of transitions used for defining the successive levels of G'. Namely, we have to ensure that the collection of levels thus defined does not contain an infinite branch that violates the fin-acceptance condition. More precisely, these levels should not include an infinite chain of edges labeled with transitions of $\mathcal{A}$ that share an acceptance condition in $\mathcal{F}$.

(Uniformization strategy and definition of G') We now give the formal inductive definition of G'. Let $V_0' \overset{def}{=} \{v_0'\}$, and let $L'(v_0') \overset{def}{=} q_I$. We assume that the acceptance conditions in the finite set $\mathcal{F}$ are totally ordered by a relation $\prec \subseteq \mathcal{F} \times \mathcal{F}$. In addition to defining the levels V_i', we shall also define a function $\widetilde{F}_i : Q \to 2^{\mathcal{F}}$ for all $0 \leq i < \omega$; intuitively, $f \in \widetilde{F}_i(q)$ shall hold for some acceptance condition $f \in \mathcal{F}$ only if the most recently defined levels of G' include a nonempty chain of edges labeled with self-loops starting from the state q, all of which have the condition f in their acceptance conditions. Let $\widetilde{F}_0(q) \overset{def}{=} \emptyset$ for all $q \in Q$. It is clear that each nonempty $\widetilde{F}_i(q)$ contains a $\prec$-minimal element.

Let $T_i : Q \to 2^\Delta$ (for each level $0 \leq i < \omega$ of G) be a function which collects the transitions of $\mathcal{A}$ that label the edges starting from a node labeled with the state q at level i of G; formally,

$$T_i(q) \overset{def}{=} \big\{L\big(\langle v, V'\rangle\big) \in \Delta \,\big|\, \exists\langle v, V'\rangle \in E \cap (V_i \times 2^{V_{i+1}}) : L(v) = q\big\}.$$

(Because G is a run, G satisfies the forward causality requirement, and thus $T_i(q) \neq \emptyset$ holds for all $0 \leq i < \omega$ and $q \in L(V_i)$.) We divide $T_i(q)$ further into three pairwise disjoint (possibly empty) partitions $T_{i,1}(q)$, $T_{i,2}(q)$ and $T_{i,3}(q)$ by defining

$$
\begin{aligned}
T_{i,1}(q) &\stackrel{\text{def}}{=} \bigl\{t \in T_i(q) \bigm| t \text{ is not a self-loop of } \mathcal{A}\bigr\},\\
T_{i,2}(q) &\stackrel{\text{def}}{=} \bigl\{t \in T_i(q) \setminus T_{i,1}(q) \bigm| t \text{ is not a } \min_\prec \widetilde{F}_i(q)\text{-transition}\bigr\}, \text{ and}\\
T_{i,3}(q) &\stackrel{\text{def}}{=} T_i(q) \setminus \bigl(T_{i,1}(q) \cup T_{i,2}(q)\bigr).
\end{aligned}
$$

(If $\widetilde{F}_i(q) = \emptyset$, then we consider no transition in $T_i(q)$ to be a $\min_\prec \widetilde{F}_i(q)$-transition in the definition of $T_{i,2}(q)$.)

Assume that V'_i and $\widetilde{F}_i$ have already been defined for some $0 \leq i < \omega$, and assume also that the labels of the nodes at level i of G' form a subset of the node labels at the corresponding level of G, i.e., $L'(V'_i) \subseteq L(V_i)$ (this clearly holds for the level V'_0 of G'). We now choose for all $q \in L'(V'_i)$ a transition $t_{i,q} = \langle q, \Gamma_{i,q}, F_{i,q}, Q'_{i,q}\rangle \in T_{i,k}(q)$, where $k \in \{1,2,3\}$ is the least index of a nonempty partition of $T_i(q)$ (because $T_i(q) \neq \emptyset$ holds, such a partition always exists). That is, we choose $t_{i,q}$ from $T_i(q)$ by preferring non-self-loops over self-loops and self-loops that are not $\min_\prec \widetilde{F}_i(q)$-transitions over other self-loops.

Write $\bigcup_{q \in L'(V'_i)} Q'_{i,q} = \{q_{i,1}, \ldots, q_{i,n_i}\}$ for some $0 \leq n_i < \omega$ (where $q_{i,j} \neq q_{i,k}$ for all $1 \leq j, k \leq n_i$, $j \neq k$). Define the level $i+1$ of G' as a set of n_i new nodes $V'_{i+1} \stackrel{\text{def}}{=} \{v_{i,1}, \ldots, v_{i,n_i}\}$, and let $L'(v_{i,j}) \stackrel{\text{def}}{=} q_{i,j}$ for all $1 \leq j \leq n_i$.

Because the labeling L is consistent in G, it follows immediately that $L'(V'_{i+1}) \subseteq L(V_{i+1})$, and thus we can repeat the same inductive construction at level $i+1$ of G' after first defining the function $\widetilde{F}_{i+1}$. For all $q \in Q$, let

$$
\widetilde{F}_{i+1}(q) \stackrel{\text{def}}{=} \begin{cases} F_{i,q} & \text{if } q \in L'(V'_i) \cap Q'_{i,q} \text{ and } \widetilde{F}_i(q) = \emptyset\\ F_{i,q} \cap \widetilde{F}_i(q) & \text{if } q \in L'(V'_i) \cap Q'_{i,q} \text{ and } \widetilde{F}_i(q) \neq \emptyset\\ \emptyset & \text{otherwise.} \end{cases}
$$

In the first two cases, the transition $t_{i,q}$ is well-defined (because $q \in L'(V'_i)$ holds), and it is a self-loop of $\mathcal{A}$ starting from the state q ($q \in Q'_{i,q}$). If $\widetilde{F}_i(q) = \emptyset$ holds, then $\widetilde{F}_i(q)$ has no $\prec$-minimal element. In this case we simply initialize $\widetilde{F}_{i+1}(q)$ with the acceptance conditions of $t_{i,q}$: clearly, $(t_{i,q})$ is then a nonempty chain of self-loops through q that trivially share all acceptance conditions in $F_{i,q}$.

Otherwise, if $\widetilde{F}_i(q) \neq \emptyset$ already contains a $\prec$-minimal element, then the most recently defined levels of G' contain a nonempty chain of edges corresponding to a chain of self-loops through q that share all acceptance conditions in $\widetilde{F}_i(q)$. The second case of the definition now guarantees that $\widetilde{F}_{i+1}(q)$ will contain a condition $f \in \widetilde{F}_i(q)$ only if also the self-loop $t_{i,q}$ includes f in its acceptance conditions. Therefore, $\widetilde{F}_{i+1}(q)$ has the intended meaning at level $i+1$ of G'.

Finally, if $q \notin L'(V'_i) \cap Q'_{i,q}$ holds, then no transition chosen from $T_i(q)$ starts or extends a chain of self-loops with source state q. In this case we define $\widetilde{F}_{i+1}(q)$ to be empty.

This completes the inductive definitions of V'_i and $\widetilde{F}_i$. To define the edges of G', we let

$$E' \stackrel{def}{=} \bigcup_{0 \le i < \omega} \left\{ \langle v, V'' \rangle \in V'_i \times 2^{V'_{i+1}} \,\middle|\, L'(V'') = Q'_{i,L'(v)} \right\}$$

and for all $e = \langle v, V'' \rangle \in E' \cap (V'_i \times 2^{V'_{i+1}})$ $(0 \le i < \omega)$, $L'(e) \stackrel{def}{=} t_{i,L'(v)}$.

(G' is a uniform run of $\mathcal{A}$ on w) We check that G' is a uniform run of $\mathcal{A}$ on w. It is easy to see that each level of G' consists of uniquely labeled nodes by construction, and thus G' satisfies the constraint required of uniform runs. It remains to be checked that G' is a run of $\mathcal{A}$ on w.

(Partitioning) It follows directly from the definitions of V' and E' that V'_0 is a singleton, V' is partitioned into finite disjoint levels, and the edges of E' lie between successive levels of G'.

(Forward causality and consistency of L') Clearly $L'(v'_0) = q_I$ holds. Let $v \in V'_i$ for some $0 \le i < \omega$. Because $L'(v) \in L'(V'_i) \subseteq L(V_i)$ holds, G contains at least one node at level i labeled with the state $L'(v)$, and because G satisfies the forward causality constraint, each such node has an outgoing edge labeled with a transition in $T_i(q)$. Therefore the transition $t_{i,q} = \langle q, \Gamma_{i,q}, F_{i,q}, Q'_{i,q} \rangle \in T_i(q)$ is well-defined (i.e., $t_{i,q} = L(e)$ for some $e \in E$). By the definition of G', $Q'_{i,q} \subseteq L'(V'_{i+1})$ holds, and thus there exists an edge $e' = \langle v, V'' \rangle \in E' \cap (V'_i \times 2^{V'_{i+1}})$. This edge is unique in E', because all nodes at level $i+1$ of G' are labeled with distinct states of $\mathcal{A}$. Furthermore, because $L'(e') = t_{i,q} = L(e)$ holds and because L is consistent, it follows that also the labeling L' is consistent.

(Backward causality) If $v' \in V'_i$ for some $1 \le i < \omega$, then there exists a state $q \in L'(V'_{i-1}) \subseteq L(V_{i-1})$ and a transition $t = \langle q, \Gamma, F, Q' \rangle \in T_{i-1}(q)$ such that $L'(v') \in Q'$ holds. Therefore, there exists a node $v \in V'_{i-1}$ and an edge $e = \langle v, V'' \rangle \in E'$ such that $L'(v) = q$ and $L'(V'') = Q'$ hold. Because no two nodes in V'_i have the same label, it follows that $v' \in V''$, and thus v' is a successor of v in G'.

We conclude that G' is a uniform run of $\mathcal{A}$ on w.

(G' is fin-accepting) If $\mathcal{F} = \emptyset$, then $\text{fin}(\beta) = \emptyset$ obviously holds for all $\beta \in \mathcal{B}(G')$, and thus G' is trivially fin-accepting.

Assume that $\mathcal{F} \neq \emptyset$ holds, and G' is not fin-accepting. Then G' contains an infinite branch $(e_i)_{0 \le i < \omega} \in \mathcal{B}(G')$ that violates the fin-acceptance condition, and there exists a minimal index $0 \le j < \omega$ such that the transitions $L'(e_i)$ $(j \le i < \omega)$ share an acceptance condition $f \in \mathcal{F}$.

On the other hand, if $\mathcal{A}$ is a self-loop alternating automaton, there exists a minimal index $0 \le k < \omega$ such that all transitions $L'(e_i)$ are self-loops of $\mathcal{A}$ having a common source state $q \in Q$ for all $k \le i < \omega$ (Proposition 2.3.18). It now follows from the definition of G' that for all $k \le i < \omega$, the transition $L'(e_i)$ is the transition chosen by the uniformization strategy from $T_i(q)$ at level i, i.e., $L'(e_i) = t_{i,q} = \langle q, \Gamma_{i,q}, F_{i,q}, Q'_{i,q} \rangle$ holds. Furthermore, because L' is consistent, $q \in L'(V'_i) \cap Q'_{i,q}$ holds for all $k \le i < \omega$.

Let $\ell \stackrel{def}{=} \max\{j, k\}$. Obviously, $f \in \bigcap_{\ell \le i < \omega} F_{i,q}$ holds by the assumption. Furthermore, there exists an index $\ell \le \ell' < \omega$ such that $\widetilde{F}_i(q) \neq \emptyset$ holds for all $\ell' \le i < \omega$: for if $\widetilde{F}_i(q) = \emptyset$ holds for some $\ell \le i < \omega$, then, because

$q \in L'(V_i') \cap Q'_{i,q}$ holds, it follows from the definition of the function $\widetilde{F}_{i+1}(q)$ that $\widetilde{F}_{i+1}(q) = F_{i,q} \ni f$ holds. Then, $\widetilde{F}_{i'}(q) \neq \emptyset$ holds by induction on i' for all $i+1 \leq i' < \omega$, because the assumption that $f \in \widetilde{F}_{i'}(q)$ holds for some $i+1 \leq i' < \omega$ implies (because of the fact that $q \in L'(V'_{i'}) \cap Q'_{i',q}$ holds) that $\widetilde{F}_{i'+1}(q) = F_{i',q} \cap \widetilde{F}_{i'}(q) \ni f$. We may thus choose $\ell' \stackrel{def}{=} i+1$ in this case.

Because $\widetilde{F}_i(q) \neq \emptyset$ holds for all $\ell' \leq i < \omega$, it is easy to see from the definition of the functions $\widetilde{F}_i(q)$ that $\widetilde{F}_{i+1}(q) \subseteq \widetilde{F}_i(q)$ holds for all $\ell' \leq i < \omega$. Because $\widetilde{F}_{\ell'}(q)$ is finite, there exists an index $\ell' \leq \ell'' < \omega$ and a nonempty set of acceptance conditions $F \subseteq \mathcal{F}$ such that $\widetilde{F}_i(q) = F$ holds for all $\ell'' \leq i < \omega$. Furthermore, $F \subseteq F_{i,q}$ also holds for all $\ell'' \leq i < \omega$. Let $f_{\min} \stackrel{def}{=} \min_{\prec} F$ be the $\prec$-minimal acceptance condition in F.

Let $\ell'' \leq i < \omega$, and let $v \in V_i$ be a node in G labeled with the state q (such a node exists, because $q \in L'(V_i') \subseteq L(V_i)$ holds). Because $t_{i,q}$ is a self-loop of $\mathcal{A}$ that is also an $f_{\min}$-transition, so is the transition labeling the edge that starts from the node v in G (otherwise the uniformization strategy would have preferred this transition when choosing a transition from $T_i(q)$). Because the labeling L is consistent, the node v has a successor in G that is labeled with the state q. By induction on i, it follows that G contains an infinite branch with an infinite suffix of edges labeled with $f_{\min}$-transitions of $\mathcal{A}$. But then G cannot be a fin-accepting run of $\mathcal{A}$ on w, which is a contradiction. It follows that G' is a uniform fin-accepting run of $\mathcal{A}$ on w.

(If) The result follows in this direction immediately because all uniform fin-accepting runs of $\mathcal{A}$ on w are obviously fin-accepting. □

4.2 NONDETERMINIZATION CONSTRUCTION

The expressive equivalence of alternating and nondeterministic automata on finite words follows already from the first results on alternation [Kozen 1976; Chandra and Stockmeyer 1976; Brzozowski and Leiss 1980; Chandra et al. 1981]. Miyano and Hayashi [1984a,b] showed that this expressive equivalence carries over to infinite words for automata working in inf-acceptance mode using a single acceptance condition associated with states of the automaton. Isli [1994, 1996] used a construction similar to the one of Miyano and Hayashi to show that an alternating automaton with n states, m of which are designated "accepting", can be translated into a nondeterministic automaton with at most $(\frac{2}{3})^m \cdot 3^n$ states. The expressive equivalence of alternating and nondeterministic automata under more general notions of acceptance and types of input was investigated (and proved) by Lindsay [1988], Emerson and Jutla [1991] and Muller and Schupp [1995].

On the other hand, the translation of alternating automata into nondeterministic automata was considered also for automata with structural constraints. Muller et al. [1986, 1992] presented a construction for translating weak alternating automata on infinite trees into nondeterministic automata working in inf-acceptance mode. Very weak automata on words were further studied by Rohde [1997], who proposed a construction for translating very weak alternating automata working on transfinite words into nondeterministic automata, and Gastin and Oddoux [2001], who showed that every very weak alternating automaton on infinite words (with n states, m of which are associated with a single inf- or fin-acceptance condition) can be translated into an equivalent nondeterministic automaton having at most 2^n states and m inf-acceptance conditions on transitions. As a matter of fact, this construction applies directly also to automata with a relaxed notion of one-weakness [Ben-David et al. 2005]; the knowledgeable reader may note the similarity of the construction with a special case of the construction of Muller et al. [1986, 1992]. Similar ideas can be found also in the construction of Fritz [2005]. Hammer et al. [2005] use a nondeterminization construction that preserves state-based acceptance; their construction is correct, however, only for a subclass of very weak alternating automata they call *simple* linear weak alternating automata.

In this section we generalize the result of Gastin and Oddoux [2001] to self-loop alternating automata with multiple fin-acceptance conditions associated with their transitions. Similarly to Gastin and Oddoux's construction, we build from a self-loop alternating automaton with n states an equivalent nondeterministic automaton with 2^n states. If the self-loop alternating automaton has m acceptance conditions, the nondeterministic automaton has at most nm acceptance conditions. This worst-case upper bound is optimal for our nondeterminization construction; special cases in which this blow-up can be avoided (such as when the alternating automaton is built from an LTL formula using the translation rules) are discussed in Sect. 4.3.

4.2.1 Universal Subset Construction

Consider a uniform run of a self-loop alternating automaton $\mathcal{A}$. Because each level of this run comprises a (possibly empty) set of nodes labeled with distinct states of $\mathcal{A}$, the labels of the nodes in the level form a subset of states of $\mathcal{A}$ with no duplicates. Intuitively, this run can be seen as a run of another automaton on the same input by collapsing each level of the run into a single node and the edges between each pair of consecutive levels into a single edge between the nodes representing the levels (see Fig. 4.1), and by defining a labeling for these nodes and edges. The nonbranching nature of the node sequence that emerges suggests that a transition labeling an edge in the sequence should not have more than one target state to ensure the consistency of the labeling. Because each uniform run of $\mathcal{A}$ can be identified in a similar way with a nonbranching sequence of nodes, it follows that the underlying automaton can actually be made nondeterministic. The states of this automaton are subsets of states of $\mathcal{A}$, and its transitions are obtained by "synchronizing" transitions of $\mathcal{A}$ starting from a given subset of states.

The above intuition for simulating self-loop alternating automata with nondeterministic automata resembles very closely the well-known *subset construction* of Rabin and Scott [1959] used for simulating nondeterministic automata on finite words with deterministic automata (i.e., automata whose every state has a unique successor on each symbol of the alphabet). In the case of alternating automata, however, the subsets consist of the current states of the active copies of the alternating automaton, instead of possible current states of a single copy of the automaton. The above intuitive construction suggests combining a set of transitions taken at a particular level

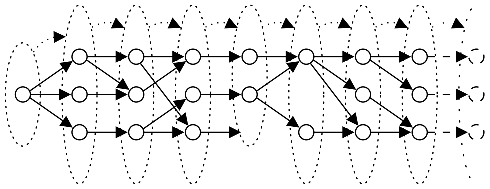


Fig. 4.1: Collapsing the levels of a uniform run of an alternating automaton into a nonbranching sequence of nodes

of a run by a collection of active copies of an alternating automaton into a transition that simulates the change effected by the transitions in the set of the active copies of the automaton. Because the labeling of the run is consistent, all of these transitions include the current input symbol in their guard. This fact suggests defining the guard of the simulating transition as the set intersection of the guards of the individual transitions taken by the copies of the alternating automaton. We sometimes refer to this principle of defining the states and transitions of a nondeterministic automaton as the *universal subset construction* for self-loop alternating automata (to distinguish it from the classic "existential" construction for nondeterministic automata on finite inputs).[2]

We have not yet considered how to define the acceptance conditions for the transitions of the nondeterministic automaton. Similarly to the target states and the guard of a transition that simulates a set of transitions, it would seem possible to define the acceptance conditions of the simulating transition as a direct combination (such as the union) of the acceptance conditions of the other transitions. This intuitive idea is not correct in the general case, however. Below, we present a construction that introduces new acceptance conditions for the nondeterministic automaton; the fact that these new acceptance conditions are indeed necessary for the universal subset construction in the general case will be shown in Sect. 4.2.3.

Theorem 4.2.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton. Define the automaton $\mathcal{A}' = \langle \Sigma, 2^Q, \Delta', \{q_I\}, Q \times \mathcal{F} \rangle$, where, for all $Q' \in 2^Q$, $\Gamma \subseteq \Sigma$, $F \subseteq Q \times \mathcal{F}$ and $\mathcal{Q} \subseteq 2^Q$,*

$$\langle Q', \Gamma, F, \mathcal{Q} \rangle \in \Delta' \quad \text{iff} \quad \text{for all } q \in Q', \text{ there exists a transition } \langle q, \Gamma_q, F_q, Q'_q \rangle \in \Delta \text{ such that } \Gamma = \bigcap_{q \in Q'} \Gamma_q,\ F = \bigcup_{q \in Q' \cap Q'_q} \big(\{q\} \times F_q\big), \text{ and } \mathcal{Q} = \big\{ \bigcup_{q \in Q'} Q'_q \big\} \text{ hold.}$$

(In particular, $\langle \emptyset, \Sigma, \emptyset, \{\emptyset\} \rangle \in \Delta'$.) The automaton $\mathcal{A}'$ is nondeterministic, and $\mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}_{\text{fin}}(\mathcal{A}')$ holds.

[2]We note that there is no corresponding "existential" subset construction for translating arbitrary nondeterministic automata on infinite words into inf- or fin-equivalent deterministic automata, because nondeterministic and deterministic automata are not expressively equivalent under these notions of acceptance (see, for example, [Vardi 1996]): in general, a more complex notion of acceptance is needed for the deterministic automaton [McNaughton 1966; Safra 1988].

Proof: It is clear from the definition that the target states of each transition of $\mathcal{A}'$ are singletons (containing a subset of Q), and thus $\mathcal{A}'$ is nondeterministic. We show that $\mathcal{A}$ and $\mathcal{A}'$ are fin-equivalent.

($\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}')$) Let $G = \langle V, E, L\rangle$ be a uniform fin-accepting run of $\mathcal{A}$ on $w \in \Sigma^\omega$. We construct a fin-accepting run $G' = \langle V', E', L'\rangle$ of $\mathcal{A}'$ on w.

(Definition of G') Let $V' \overset{def}{=} \bigcup_{0\leq i<\omega}\{v'_i\}$ (with $V'_i \overset{def}{=} \{v'_i\}$ for all $0 \leq i < \omega$), $E' \overset{def}{=} \bigcup_{0\leq i<\omega}\big\{\langle v'_i, V'_{i+1}\rangle\big\}$, and $L'(v'_i) \overset{def}{=} L(V_i)$ for all $0 \leq i < \omega$. To define the label of the edge starting from the node v'_i ($0 \leq i < \omega$), we first write $V_i = \{v_{i,1}, \ldots, v_{i,n_i}\}$ for some $0 \leq n_i < \omega$ (where $v_{i,j} \neq v_{i,k}$ holds for all $1 \leq j, k \leq n_i$, $j \neq k$). Because G is uniform, $L(v_{i,j}) = q_{i,j} \neq q_{i,k} = L(v_{i,k})$ holds for all $1 \leq j, k \leq n_i$ ($j \neq k$), and thus there are n_i distinct transitions $T_i \subseteq \Delta$ labeling edges starting from the nodes in V_i:

$$T_i \overset{def}{=} \bigcup_{1\leq j\leq n_i} \big\{L\big(\langle v_{i,j}, V''\rangle\big) \,\big|\, \langle v_{i,j}, V''\rangle \in E\big\} = \bigcup_{1\leq j\leq n_i} \big\{\langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j}\rangle\big\}.$$

The label of the edge $\langle v'_i, V'_{i+1}\rangle$ ($0 \leq i < \omega$) is then given by $L'\big(\langle v'_i, V'_{i+1}\rangle\big) \overset{def}{=} \big\langle \bigcup_{1\leq j\leq n_i}\{q_{i,j}\}, \bigcap_{1\leq j\leq n_i}\Gamma_{i,j}, \bigcup_{\substack{1\leq j\leq n_i\\ q_{i,j}\in Q'_{i,j}}} \big(\{q_{i,j}\} \times F_{i,j}\big), \big\{\bigcup_{1\leq j\leq n_i} Q'_{i,j}\big\}\big\rangle$.

(G' is a run of $\mathcal{A}'$ on w) We check that the graph G' satisfies all constraints required of a run of $\mathcal{A}'$ on w.

(Partitioning) Obviously $V'_0 = \{v'_0\}$ is a singleton, and G' is partitioned into finite disjoint levels with edges between consecutive levels.

(Causality) Clearly, each level of G' contains only one node, and for all $0 \leq i < \omega$, the node $v'_i \in V'_i$ has the unique outgoing edge $\langle v'_i, V'_{i+1}\rangle \in E'$. On the other hand, the node $v'_i \in V'_i$ ($1 \leq i < \omega$) is a successor of the node $v'_{i-1} \in V'_{i-1}$.

(Consistency of L') It is clear from the definition that $L'(v'_0) = L(V_0) = L\big(\{v_0\}\big) = \{q_I\}$ holds. Let $0 \leq i < \omega$, and let $v = v'_i \in V'_i$. Because $T_i \subseteq \Delta$ holds, it follows from the definition of $\mathcal{A}'$ that Δ' contains the transition $\big\langle \bigcup_{1\leq j\leq n_j}\{q_{i,j}\}, \bigcap_{1\leq j\leq n_i}\Gamma_{i,j}, \bigcup_{\substack{1\leq j\leq n_i\\ q_{i,j}\in Q'_{i,j}}} \big(\{q_{i,j}\}\times F_{i,j}\big), \big\{\bigcup_{1\leq j\leq n_i} Q'_{i,j}\big\}\big\rangle = L'\big(\langle v'_i, V'_{i+1}\rangle\big)$. Because G is a run, $w(i) \in \Gamma_{i,j}$ holds for all $1 \leq j \leq n_i$, and thus $w(i) \in \bigcap_{1\leq j\leq n_i}\Gamma_{i,j}$ holds. Furthermore, because L is consistent, it follows that $\bigcup_{1\leq j\leq n_i}\{q_{i,j}\} = \bigcup_{1\leq j\leq n_i}\big\{L(v_{i,j})\big\} = L(V_i) = L'(v'_i)$ and $\big\{\bigcup_{1\leq j\leq n_i} Q'_{i,j}\big\} = \big\{L(V_{i+1})\big\} = \big\{L(v'_{i+1})\big\} = L(V'_{i+1})$ hold. Therefore also the labeling L' is consistent.

We conclude that G' is a run of $\mathcal{A}'$ on w.

(G' is fin-accepting) Suppose that G' is not fin-accepting. Then there exists an index $0 \leq j < \omega$ and an acceptance condition $\langle q, f\rangle \in Q \times \mathcal{F}$ such that for all $j \leq i < \omega$, the transition $L'\big(\langle v'_i, V'_{i+1}\rangle\big) \in \Delta'$ is a $\langle q, f\rangle$-transition. Let $j \leq i < \omega$. By the definition of L', T_i contains an f-transition $t \in \Delta$ (with source state q) that is a self-loop of $\mathcal{A}$. Because t is the consistent label of an edge starting from a node $v \in V_i$ with $L(v) = q$, v has a successor in V_{i+1} that is also labeled with the state q. Furthermore, this successor is the only node in V_{i+1} that has q as its label, because G is uniform. By induction on i, it follows that G contains an infinite branch with an infinite

suffix of edges labeled with f-transitions. This contradicts the assumption that G is fin-accepting. Therefore G' is a fin-accepting run of $\mathcal{A}'$ on w, and $\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}')$ holds.

($\mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A})$) Let $G' = \langle V', E', L' \rangle$ be a fin-accepting run of $\mathcal{A}'$ on $w \in \Sigma^\omega$. Because $\mathcal{A}'$ is nondeterministic, we may assume (without loss of generality) that G' is uniform (Proposition 4.1.1). Therefore, every level V'_i of G' consists of a single node v'_i for all $0 \leq i < \omega$. Write $L'(v'_i) = \{q_{i,1}, \ldots, q_{i,n_i}\} \in 2^Q$ for some $0 \leq n_i < \omega$ (where $q_{i,j} \neq q_{i,k}$ holds for all $1 \leq j, k \leq n_i$, $j \neq k$) for all $0 \leq i < \omega$. Similarly to the other direction, we define a fin-accepting run $G = \langle V, E, L \rangle$ of $\mathcal{A}$ on w.

(Definition of G) For all $0 \leq i < \omega$, let V_i consist of n_i new nodes $V_i \overset{\text{def}}{=} \{v_{i,1}, \ldots, v_{i,n_i}\}$, and let $L(v_{i,j}) \overset{\text{def}}{=} q_{i,j}$ for all $1 \leq j \leq n_i$. Clearly, no two nodes in V_i have the same label, and $L(V_i) = L'(v'_i)$ holds for all $0 \leq i < \omega$.

Because G' is a uniform run of a nondeterministic automaton, the node $v'_i \in V'$ has the unique outgoing edge $\langle v'_i, \{v'_{i+1}\} \rangle \in E'$ labeled with a transition $\langle L'(v'_i), \Gamma_i, F_i, \{L'(v'_{i+1})\} \rangle \in \Delta'$ for some $\Gamma_i \subseteq \Sigma$ and $F_i \subseteq Q \times \mathcal{F}$ for all $0 \leq i < \omega$. By the definition of $\mathcal{A}'$, there exist transitions $t_{i,j} = \langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j} \rangle \in \Delta$ for all $1 \leq j \leq n_i$ such that $\Gamma_i = \bigcap_{1 \leq j \leq n_i} \Gamma_{i,j}$, $F_i = \bigcup_{\substack{1 \leq j \leq n_i \\ q_{i,j} \in Q'_{i,j}}} \big(\{q_{i,j}\} \times F_{i,j}\big)$ and $\{L'(v'_{i+1})\} = \{\bigcup_{1 \leq j \leq n_i} Q'_{i,j}\}$ hold.

Let $v \in V_i$ for some $0 \leq i < \omega$; thus $L(v) = q_{i,j} \in L'(v'_i)$ for some $1 \leq j \leq n_i$. We now define $e_{i,j} \overset{\text{def}}{=} \langle v, V'' \rangle \in V_i \times 2^{V_{i+1}}$ by taking V'' to be the unique subset of V_{i+1} for which $L(V'') = Q'_{i,j}$ holds (such a subset exists, because $Q'_{i,j} \subseteq L'(v'_{i+1}) = L(V_{i+1})$ holds, and it is unique, because all nodes in V_{i+1} have different labels). We then define $E \overset{\text{def}}{=} \bigcup_{0 \leq i < \omega} \{e_{i,1}, \ldots, e_{i,n_i}\}$ and $L(e_{i,j}) \overset{\text{def}}{=} t_{i,j}$ for all $0 \leq i < \omega$ and $1 \leq j \leq n_i$.

(G is a run of $\mathcal{A}$ on w) We check that G is a run of $\mathcal{A}$ on w.

(Partitioning) Because $L'(v'_0) = \{q_I\}$, $n_0 = 1$ holds, and thus V_0 is a singleton. Because the nodes of G' are labeled with subsets of the finite set Q, V_i is finite for all $0 \leq i < \omega$ (and disjoint from all other levels of G by definition). By the definition of E, G has edges only between its consecutive levels.

(Causality) Let $v = v_{i,j} \in V_i$ for some $0 \leq i < \omega$ and $1 \leq j \leq n_i$. By the definition of G, v has the unique outgoing edge $e_{i,j} = \langle v, V'' \rangle \in E$.

On the other hand, if $v' \in V_i$ holds for some $1 \leq i < \omega$, then $L(v') = q' \in L'(v'_i)$ holds. Because G' is a uniform run of a nondeterministic automaton, v'_i is a successor of the node $v'_{i-1} \in V'$. Let $t' \in \Delta'$ be the transition labeling the edge between these nodes in G'. Because $L'(v'_i) \neq \emptyset$, it follows from the definition of $\mathcal{A}'$ that $L'(v'_{i-1}) \neq \emptyset$, and there exists a transition $t \in \Delta$ (with source state $q \in L'(v'_{i-1})$) which is a "component" in the transition t' and includes q' in its target states. By the definition of G, V_{i-1} now contains a node v with $L(v) = q$, and the edge starting from this node is labeled with the transition t. Thus v has a successor labeled with the state q'. This successor is now necessarily the node v', because the labels of nodes in V_i are distinct by definition.

(Consistency of L) Clearly $L(V_0) = \{q_I\}$ holds. Let $v = v_{i,j} \in V$ for some $0 \leq i < \omega$ and $1 \leq j \leq n_i$. By the definition of G, the edge $e_{i,j} \in E$ starting from this node is labeled with the transition $t_{i,j} = \langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j} \rangle \in \Delta$. Clearly, $L(v) = q_{i,j}$ holds, and $L(V'') = Q'_{i,j}$ holds by the definition of $e_{i,j}$. Furthermore, because G' is a run, $w(i) \in \Gamma_i = \bigcap_{1 \leq k \leq n_i} \Gamma_{i,k}$ holds. In particular, $w(i) \in \Gamma_{i,j}$ holds, and thus the labeling $\bar{L}$ is consistent.

(G is fin-accepting) Assume that G is not fin-accepting. Then there exists an index $0 \leq j < \omega$, an acceptance condition $f \in \mathcal{F}$ and an infinite branch $(e_i)_{0 \leq i < \omega} \in \mathcal{B}(G)$ such that $L(e_i)$ is an f-transition for all $j \leq i < \omega$. Because $\mathcal{A}$ is a self-loop alternating automaton, we may choose j large enough such that the transitions $L(e_i)$ are self-loops of $\mathcal{A}$ (with source state $q \in Q$) for all $j \leq i < \omega$ (Proposition 2.3.18). Because $e_i \in E$ holds for all $j \leq i < \omega$, then $L(e_i) = t_{i,k} = \langle q, \Gamma_{i,k}, F_{i,k}, Q'_{i,k} \rangle$ holds for some $1 \leq k \leq n_i$. But then, because $t_{i,k}$ is a self-loop and $f \in F_{i,k}$ holds, it follows from the definition of $\mathcal{A}'$ that the transition $L'\big(\langle v'_i, \{v'_{i+1}\} \rangle\big)$ is a $\langle q, f \rangle$-transition for all $j \leq i < \omega$, which contradicts the assumption that G' is fin-accepting. Therefore G is a fin-accepting run of $\mathcal{A}$ on w, and $\mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A})$. $\square$

4.2.2 Number of States and Transitions in a Nondeterministic Automaton

It is easy to see that a nondeterministic automaton built from a self-loop alternating automaton with n states using the construction in Theorem 4.2.1 has at most 2^n states. By combining the translation from LTL to self-loop alternating automata (Sect. 3.1) with the construction, we obtain the following simple corollary on the complexity of translation of linear temporal logic formulas into nondeterministic automata. This upper bound for the number of states is essentially the same as in the translations of Couvreur [1999] and Gastin and Oddoux [2001] with the exception of an additive constant (that arises because our automata have a unique initial state).

Corollary 4.2.2 *Let $\varphi \in LTL(AP)$ be any LTL formula built from the elements of AP, the Boolean constants $\top$ and $\bot$, and the connectives $\{\neg, \vee, \wedge, \mathsf{X}, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$. The language of the formula φ can be recognized by a nondeterministic automaton (working on the alphabet 2^{AP}) with at most $1 + 2^{|\mathrm{Temp}([\varphi]^{\mathrm{PNF}})|} \leq 1 + 2^{2 \cdot |\mathrm{Temp}(\varphi)|} = 1 + 4^{|\mathrm{Temp}(\varphi)|}$ states ($1 + 2^{|\mathrm{Temp}(\varphi)|}$ states, if φ itself is in positive normal form). (If φ is a binary pure temporal formula, the upper bound reduces to $2^{|\mathrm{Temp}([\varphi]^{\mathrm{PNF}})|}$ states.)*

Proof: Let φ be an LTL formula in the given form. By Corollary 3.2.2, there exists a self-loop alternating automaton $\mathcal{A}$ (working on the alphabet 2^{AP}) with at most $1 + \big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big|$ states (or at most $\big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big|$ states if φ is a binary pure temporal formula) such that $\mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}(\varphi)$ holds, and this automaton can be built using the translation rules presented in Sect. 3.1. By the construction in Theorem 4.2.1, there exists a nondeterministic automaton $\mathcal{A}'$ that fin-accepts the same language. Because the subautomaton rooted at the initial state of this automaton has at most $2^{|Q|}$ states (where Q is the state set of $\mathcal{A}$), the result now follows immediately if φ is a binary pure temporal formula.

If φ is not a binary pure temporal formula, then the construction presented in Theorem 4.2.1 yields a nondeterministic automaton with at most

$2^{1+|\mathrm{Temp}([\varphi]^{\mathrm{PNF}})|}$ states. In this case, however, the alternating automaton $\mathcal{A}$ has no self-loops starting from its initial state q_I; furthermore, because $\mathcal{A}$ is a self-loop alternating automaton, no other transition of $\mathcal{A}$ can have q_I as its target state, either. It follows from the definition of $\mathcal{A}'$ that the subautomaton rooted at the initial state of $\mathcal{A}'$ contains no states corresponding to non-singleton subsets of Q that include the state q_I. Therefore, the upper bound for the size of the subautomaton reduces to $1 + 2^{|\mathrm{Temp}([\varphi]^{\mathrm{PNF}})|}$ states, and the result follows. $\square$

We mention here that in the general case, the smallest nondeterministic automaton (working on the alphabet 2^{AP}) which recognizes the language of an LTL formula (over the atomic propositions AP) always has exponentially many states in the number of pure temporal subformulas of the formula. For example [Wolper 2001], it is possible to express the behavior of an n-bit binary counter that resets itself infinitely often as an LTL formula over n atomic propositions with $O(n)$ pure temporal subformulas (hence, as a self-loop alternating automaton with $O(n)$ states), but a corresponding nondeterministic automaton working on the same alphabet has no less than 2^n states.

It is easy to see from the construction in Theorem 4.2.1 that every transition of the nondeterministic automaton built from a self-loop alternating automaton is defined in terms of a subset of transitions of the self-loop alternating automaton. Therefore, the nondeterministic automaton built from a self-loop alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ cannot have more than $2^{|\Delta|}$ transitions. Combining this upper bound directly with the exponential ($2^{O(|\varphi|)}$) upper bound for the number of transitions in a self-loop alternating automaton built from an LTL formula $\varphi \in LTL(AP)$ using the translation rules presented in Sect. 3.1 (see Sect. 3.2.2) yields a doubly exponential upper bound (in $|\varphi|$) for the number of transitions in a nondeterministic automaton built for the formula φ. The number of transitions can nevertheless be shown to be, in effect, only singly exponential in $|\varphi|$; the details follow in Sect. 4.3.4.

4.2.3 Number of Acceptance Conditions

Given a self-loop alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$, the construction of Theorem 4.2.1 gives a simple upper bound ($|Q| \cdot |\mathcal{F}|$) for the number of acceptance conditions in a fin-equivalent nondeterministic automaton built from $\mathcal{A}$ by applying the universal subset construction. A more accurate upper bound can be found by observing that the nondeterministic automaton built from $\mathcal{A}$ using the construction has a $\langle q, f \rangle$-transition (for some $\langle q, f \rangle \in Q \times \mathcal{F}$) only if $q \in Q$ is an f-state of $\mathcal{A}$. Because the acceptance conditions for which no corresponding transitions exist do not affect fin-acceptance (clearly, none of these conditions can appear in the label of an edge in any infinite branch of a run of the automaton), these conditions can be safely removed from the nondeterministic automaton without changing its language. A more accurate upper bound for the number of acceptance conditions needed for the nondeterministic automaton is thus given by the equation

$$\sum_{f \in \mathcal{F}} |\{q \in Q : q \text{ is an } f\text{-state of } \mathcal{A}\}| \leq |Q| \cdot |\mathcal{F}|.$$

The $|Q| \cdot |\mathcal{F}|$ upper bound in nevertheless tight if the states and transitions of the nondeterministic automaton are to be defined using the universal subset construction.

Proposition 4.2.3 *A nondeterministic automaton built from a self-loop alternating automaton with $1 \le n < \omega$ states and $1 \le m < \omega$ acceptance conditions (and an alphabet of at least nm symbols) using the universal subset construction needs nm acceptance conditions to recognize the language of the alternating automaton in the worst case.*

<u>*Proof:*</u> Let Σ be a finite alphabet, and let $w \in \Sigma^\omega$. We use the notation $\mathrm{occ}(w) \stackrel{\mathit{def}}{=} \{\sigma \in \Sigma \mid w(i) = \sigma \text{ for some } 0 \le i < \omega\}$ for the set of symbols occurring in w, and $\inf(w) \stackrel{\mathit{def}}{=} \{\sigma \in \Sigma \mid w(i) = \sigma \text{ for infinitely many } 0 \le i < \omega\}$ for the set of symbols which occur in w infinitely many times.

Let $\{\Sigma_{n,m}\}_{1 \le n,m < \omega}$ be a family of alphabets, where the alphabet $\Sigma_{n,m}$ $(1 \le n, m < \omega)$ is the union of n pairwise disjoint m-symbol alphabets $\Sigma'_{i,m}$ $(1 \le i \le n)$, i.e., $\Sigma_{n,m} \stackrel{\mathit{def}}{=} \bigcup_{i=1}^{n} \Sigma'_{i,m}$, where $\Sigma'_{i,m} \stackrel{\mathit{def}}{=} \{\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,m}\}$, and $\Sigma'_{i,m} \cap \Sigma'_{j,m} = \emptyset$ holds for all $1 \le i, j \le n$, $i \ne j$. Define also the corresponding family of languages $\{\mathcal{L}_{n,m}\}_{1 \le n,m < \omega}$, where, for all $1 \le n, m < \omega$,

$$\mathcal{L}_{n,m} \stackrel{\mathit{def}}{=} \big\{ w \in \Sigma_{n,m}^\omega \,\big|\, \Sigma'_{1,m} \subseteq \inf(w), \text{ and if } \Sigma'_{i,m} \cap \mathrm{occ}(w) \ne \emptyset \text{ for some } 2 \le i < \omega, \text{ then also } \Sigma'_{i,m} \subseteq \inf(w) \big\}.$$

Let $1 \le n, m < \omega$ be fixed. We show that the language $\mathcal{L}_{n,m}$ can be fin-recognized by a self-loop alternating automaton (on the alphabet $\Sigma_{n,m}$) with n states and m acceptance conditions, but a nondeterministic automaton built from the alternating automaton using the universal subset construction needs at least nm acceptance conditions to fin-recognize the same language.

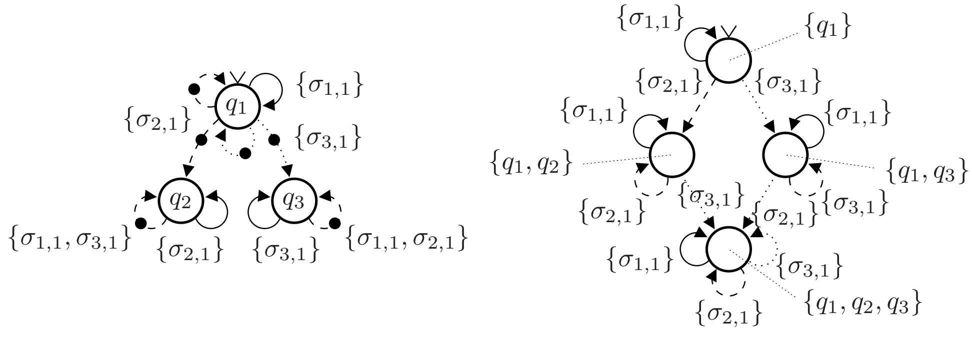
Let $2 \le i \le n$. It is easy to check that the language $\big\{ w \in \Sigma_{n,m}^\omega \,\big|\, \Sigma'_{i,m} \subseteq \inf(w) \big\}$ is fin-recognized by the single-state automaton $\mathcal{A}_i$ with $m+1$ self-loops and m acceptance conditions $\mathcal{F}_m \stackrel{\mathit{def}}{=} \{f_1, f_2, \ldots, f_m\}$; formally, $\mathcal{A}_i \stackrel{\mathit{def}}{=} \big\langle \Sigma_{n,m}, \{q_i\}, \Delta_i, q_i, \mathcal{F}_m \big\rangle$, where

$$\Delta_i \stackrel{\mathit{def}}{=} \big\{ \big\langle q_i, \{\sigma_{i,j}\}, \mathcal{F}_m \setminus \{f_j\}, \{q_i\} \big\rangle \,\big|\, 1 \le j \le m \big\} \cup \big\{ \big\langle q_i, \Sigma_{n,m} \setminus \Sigma'_{i,m}, \mathcal{F}_m, \{q_i\} \big\rangle \big\}.$$

Clearly, for all $\sigma \in \Sigma_{n,m}$, there is a unique transition in Δ_i that includes σ in its guard.

We now use the automata $\mathcal{A}_i$ $(2 \le i < \omega)$ to build an automaton $\mathcal{A}_{n,m}$ that fin-recognizes the language $\mathcal{L}_{n,m}$. Similarly to the automata $\mathcal{A}_i$, $\mathcal{A}_{n,m}$ has an initial state q_1 used for checking that the input of the automaton contains infinitely many occurrences of each symbol in $\Sigma'_{1,m}$. Whenever the automaton reads a symbol from some $\Sigma'_{i,m}$ $(2 \le i < \omega)$, it spawns a copy of the automaton $\mathcal{A}_i$ to check that the input contains also infinitely many occurrences of each symbol in $\Sigma'_{i,m}$. The automaton always keeps a copy of itself in its initial state. Formally, $\mathcal{A}_{n,m} \stackrel{\mathit{def}}{=} \langle \Sigma_{n,m}, Q_n, \Delta_{n,m}, q_1, \mathcal{F}_m \rangle$, where

$$\begin{aligned} Q_n &\stackrel{\mathit{def}}{=} \{q_1, q_2, \ldots, q_n\} \qquad \text{and} \\ \Delta_{n,m} &\stackrel{\mathit{def}}{=} \big\{ \big\langle q_1, \{\sigma_{1,i}\}, \mathcal{F}_m \setminus \{f_i\}, \{q_1\} \big\rangle \,\big|\, 1 \le i \le m \big\} \\ &\quad \cup \big\{ \big\langle q_1, \Sigma'_{i,m}, \mathcal{F}_m, \{q_1, q_i\} \big\rangle \,\big|\, 2 \le i \le n \big\} \\ &\quad \cup \textstyle\bigcup_{i=2}^{n} \Delta_i. \end{aligned}$$



(a) (b)

Fig. 4.2: Examples of automata used in the proof of Proposition 4.2.3. (a) The automaton $\mathcal{A}_{3,1}$. (b) State–transition structure of the nondeterministic automaton obtained from $\mathcal{A}_{3,1}$ using the universal subset construction (transitions with empty guards and states not reachable from the initial state omitted)

It is easy to see that $\mathcal{A}_{n,m}$ is a self-loop alternating automaton, because there are no transitions from any state $q_i \in Q_n$ $(2 \leq i \leq n)$ to another state $q \in Q_n \setminus \{q_i\}$. For illustration, see Fig. 4.2 (a) for the automaton $\mathcal{A}_{3,1}$.

Observe that, for all $q \in Q_n$ and $\sigma \in \Sigma_{n,m}$, $\Delta_{n,m}$ again contains a unique transition with source state q and σ in its guard. Thus the automaton has a run on every $w \in \Sigma_{n,m}^{\omega}$, and the run on w is unique with respect to the sets of states and transitions labeling the nodes at the levels and the edges starting from the levels, respectively. Obviously, the run always contains an infinite branch that stays in the state q_1, and if $\Sigma'_{i,m} \cap \mathrm{occ}(w) \neq \emptyset$ holds for some $2 \leq i \leq n$, the run contains also an infinite branch that converges to the state q_i. It is therefore easy to see that the run is fin-accepting only if $w \in \mathcal{L}_{n,m}$ holds. On the other hand, it is straightforward to check that all runs of $\mathcal{A}_{n,m}$ on w are fin-accepting if $w \in \mathcal{L}_{n,m}$, and thus the automaton $\mathcal{A}_{n,m}$ fin-recognizes the language $\mathcal{L}_{n,m}$.

Let $\mathcal{A}'_{n,m}$ be the nondeterministic automaton on the alphabet $\Sigma_{n,m}$ obtained from $\mathcal{A}_{n,m}$ by defining its states 2^{Q_n} and transitions $\Delta'_{n,m}$ using the universal subset construction, where we deliberately give each transition of $\mathcal{A}'_{n,m}$ an empty set of acceptance conditions (see Fig. 4.2 (b) for illustration). Observe again that for every subset $Q' \subseteq Q_n$ and every symbol $\sigma \in \Sigma_{n,m}$, $\mathcal{A}'_{n,m}$ has a unique transition $\langle Q', \Gamma, \emptyset, \{Q''\} \rangle \in \Delta'_{n,m}$ $(\Gamma \subseteq \Sigma_{n,m}$, $Q'' \subseteq Q_n)$ such that $\sigma \in \Gamma$ holds. In particular, $\mathcal{A}'_{n,m}$ contains the transition chain $\big(\big\langle \{q_1, \ldots, q_i\}, \Sigma_m^{i+1}, \emptyset, \big\{\{q_1, \ldots, q_i, q_{i+1}\}\big\}\big\rangle\big)_{1 \leq i \leq n-1}$. Furthermore, it is straightforward to check from the construction that for all $\sigma \in \Sigma_{n,m}$, $\Delta'_{n,m}$ contains the transition $\langle Q_n, \{\sigma\}, \emptyset, \{Q_n\} \rangle$ (and these are the only transitions with a nonempty guard starting from the state Q_n).

Assume that there is a way to define the acceptance conditions of the transitions of $\mathcal{A}'_{n,m}$ as subsets of a set $\mathcal{F}$ with less than nm elements such that $\mathcal{A}'_{n,m}$ fin-recognizes the language $\mathcal{L}_{n,m}$. Let $u = \sigma_{1,1}\sigma_{2,1} \cdots \sigma_{n,1}$. It follows by the above discussion that the nodes at any level greater than or equal to n in a run of $\mathcal{A}'_{n,m}$ on an infinite word of the form uv $(v \in \Sigma_{n,m}^{\omega})$ are all labeled with the state Q_n. Because u contains a symbol from each $\Sigma'_{i,m}$, $uv \in \mathcal{L}_{n,m}$ holds only if v contains infinitely many occurrences of each symbol in $\Sigma_{n,m}$. Therefore, $\mathcal{A}'_{n,m}$ fin-accepts the language $\mathcal{L}_{n,m}$ only if the subautomaton $(\mathcal{A}'_{n,m})^{Q_n}$ fin-recognizes the language of infinite words containing infinitely many occurrences of each symbol of the alphabet $\Sigma_{n,m}$.

The language fin-accepted by $(\mathcal{A}'_{n,m})^{Q_n}$ is obviously empty if all self-loops $\langle Q_n, \{\sigma\}, \emptyset, \{Q_n\}\rangle \in \Delta'_{n,m}$ ($\sigma \in \Sigma_{n,m}$) are f-transitions for some $f \in \mathcal{F}$. Thus there exists a set of at most $|\mathcal{F}|$ self-loops that includes for each $f \in \mathcal{F}$ a self-loop that is not an f-transition. Because the guards of these self-loops are singleton subsets of $\Sigma_{n,m}$, it follows that $\mathcal{A}'_{n,m}$ fin-accepts all words of the form uw^ω for some permutation w of the symbols in the guards of these self-loops. But then, because $w = |\mathcal{F}| < nm$ holds, uw^ω contains only finitely many occurrences of some symbol $\sigma \in \Sigma_{n,m}$. This contradicts the assumption that the automaton $\mathcal{A}'_{n,m}$ fin-recognizes the language $\mathcal{L}_{n,m}$. Therefore, $\mathcal{F}$ must have at least nm conditions (which is also sufficient by Theorem 4.2.1). $\square$

4.3 AUTOMATA WITH ACCEPTANCE SYNCHRONIZED RUNS

The universal subset construction provides an intuitive way for translating self-loop alternating automata into fin-equivalent nondeterministic automata due to its resemblance to the classic existential subset construction for nondeterministic finite word automata. The construction is also essentially optimal with respect to the blow-up in the number of states in the automaton. However, as shown in Sect. 4.2.3, translating a self-loop alternating automaton with n states into an equivalent nondeterministic one using the construction may necessitate an n-fold increase in the number of the automaton's acceptance conditions. Consequently, the procedure obtained by combining the rule-based translation of LTL into self-loop alternating automata (Sect. 3.1) with the nondeterminization construction of Theorem 4.2.1 maps a formula with n syntactically distinct pure temporal subformulas (m of which are binary temporal subformulas) into a nondeterministic automaton with $O(nm)$ acceptance conditions. However, most procedures for translating LTL into nondeterministic automata with multiple inf- or fin-acceptance conditions (beginning already with the procedure of Gerth et al. [1995] and, in particular, including that of Gastin and Oddoux [2001]) manage to do the translation using $O(m)$ conditions for the nondeterministic automaton. The inferior worst-case performance of the proposed construction appears to be a consequence of associating acceptance with the transitions instead of the states of the automata.

In this section, we shall introduce a subclass of alternating automata that can be translated into nondeterministic automata without any blow-up in the number of acceptance conditions. The subclass is characterized by a combination of structural and semantic properties that guarantee the existence of a special kind of accepting runs that allows simplifying the universal subset construction. It occurs that the self-loop alternating automata built from LTL formulas using the translation rules presented in Sect. 3.1 trivially belong to this subclass of automata. Therefore, these automata can be translated into nondeterministic automata with at most as many acceptance conditions as in automata built using previously known constructions. In principle, our result is analogous to that of Hammer et al. [2005]: whereas they identify a subclass of automata which can be translated into nondeterministic au-

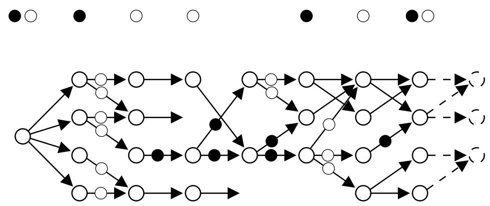


Fig. 4.3: Principle of fin-acceptance synchronization. A level labeled on top of the figure with the symbol • (resp. ◦) contains no edges labeled with the acceptance condition • (◦); a fin-acceptance synchronized run has infinitely many such levels for both conditions

tomata without giving up state-based acceptance, our subclass of self-loop alternating automata (with transition-based acceptance) admits translation into nondeterministic automata without introducing new acceptance conditions. Furthermore, similarly to the simple linear weak alternating automata of Hammer et al. [2005], our subclass is not less expressive than the class of self-loop alternating automata since its expressiveness captures all of LTL.

4.3.1 Acceptance Synchronicity

As seen in Sect. 4.1, the question on the existence of an accepting run for a self-loop alternating automaton can be answered by considering only the uniform runs of the automaton. The key to optimizing the nondeterminization construction of Sect. 4.2 is to identify an even smaller subset of runs that satisfy certain requirements on the occurrence of transitions associated with acceptance conditions. We define this subset formally as follows.

Let $G = \langle V, E, L \rangle$ be a run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ on an infinite word $w \in \Sigma^\omega$ over the alphabet Σ (where V consists of disjoint finite levels V_i as usual), and let $f \in \mathcal{F}$ be an acceptance condition. We say that the run G is *fin-synchronized with respect to f* iff there exist infinitely many $0 \leq i < \omega$ such that no transition that labels an edge starting from a node at level i of G is an f-transition of $\mathcal{A}$. The run G is *fin-acceptance synchronized* iff G is fin-synchronized with respect to all acceptance conditions in $\mathcal{F}$. (The definition of inf-synchronicity with respect to a condition $f \in \mathcal{F}$ is analogous: we simply require that G contain infinitely many levels, all edges starting from which *are* labeled with f-transitions. As before, we nevertheless consider only fin-acceptance in the following to simplify the discussion.)

Intuitively, if a fin-acceptance synchronized run of $\mathcal{A}$ on w is interpreted (in the standard way) as a description of the behavior of $\mathcal{A}$'s copies working in parallel on the input w, then, although every copy of the automaton works independently of the other copies, the copies nevertheless "cooperate" in this run with respect to every acceptance condition $f \in \mathcal{F}$ by avoiding f-transitions at certain (infinitely many) positions of the input (see Fig. 4.3 for illustration). It is easy to see that every fin-acceptance synchronized run of $\mathcal{A}$ is a fin-accepting run of $\mathcal{A}$.

Proposition 4.3.1 *Let $G = \langle V, E, L \rangle$ be a run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ on some $w \in \Sigma^\omega$. If G is fin-acceptance synchronized,* *then $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds.*

Proof: If $\mathcal{B}(G) = \emptyset$, then the claim holds trivially. Otherwise let $\beta = (e_i)_{0\leq i<\omega} \in \mathcal{B}(G)$ ($e_i \in E \cap (V_i \times 2^{V_{i+1}})$) be an infinite branch in G. We show that $\text{fin}(\beta) = \emptyset$ holds, that is, $f \notin \text{fin}(\beta)$ holds for all acceptance conditions $f \in \mathcal{F}$. This is clear for the condition $f \in \mathcal{F}$ if no edge in β is labeled with an f-transition; otherwise, if $L(e_j)$ is an f-transition for some $0 \leq j < \omega$, then, because G is fin-acceptance synchronized, there exists an index $j < k < \omega$ such that no edge in $E \cap (V_k \times 2^{V_{k+1}})$ is labeled with an f-transition. In particular, this implies that $L(e_k)$ is not an f-transition. Because j is arbitrary, β contains infinitely many edges not labeled with an f-transition, and thus $f \notin \text{fin}(\beta)$ holds. Because the same reasoning applies to all acceptance conditions, it follows that $\text{fin}(\beta) = \emptyset$ holds, and thus β is fin-accepting. Because β is arbitrary, it follows that G is a fin-accepting run of $\mathcal{A}$ on w. □

4.3.2 A Simplified Nondeterminization Construction

By Proposition 4.3.1, an alternating automaton $\mathcal{A}$ working on infinite words over the alphabet Σ fin-accepts a word $w \in \Sigma^\omega$ whenever $\mathcal{A}$ has a fin-acceptance synchronized run on w. The same result clearly holds in particular for uniform fin-acceptance synchronized runs on w. If $\mathcal{A}$ has also the converse property, i.e., if $\mathcal{A}$ has a uniform fin-acceptance synchronized run on all words $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$, then $\mathcal{A}$ can be translated into a fin-equivalent nondeterministic automaton that uses the same number of acceptance conditions as $\mathcal{A}$. Such an automaton can again be built using the universal subset construction. Actually, $\mathcal{A}$ need not even be a self-loop alternating automaton (however, it must have uniform fin-accepting runs in the sense of Sect. 4.1).

Theorem 4.3.2 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton, and assume that, for all $w \in \Sigma^\omega$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds iff $\mathcal{A}$ has a uniform fin-acceptance synchronized run on w. Define the automaton $\mathcal{A}' = \langle \Sigma, 2^Q, \Delta', \{q_I\}, \mathcal{F} \rangle$, where, for all $Q' \in 2^Q$, $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $\mathcal{Q} \subseteq 2^Q$,*

$$\langle Q', \Gamma, F, \mathcal{Q} \rangle \in \Delta' \quad \text{iff} \quad \text{for all } q \in Q', \text{ there exists a transition } \langle q, \Gamma_q, F_q, Q'_q \rangle \in \Delta \text{ such that } \Gamma = \bigcap_{q\in Q'} \Gamma_q,\ F = \bigcup_{q\in Q'} F_q, \text{ and } \mathcal{Q} = \Big\{ \bigcup_{q\in Q'} Q'_q \Big\} \text{ hold.}$$

The automaton $\mathcal{A}'$ is nondeterministic, and $\mathcal{L}_{\text{fin}}(\mathcal{A}') = \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds.

Proof: The proof is similar to that of Theorem 4.2.1 (p. 76); clearly, the definition of $\mathcal{A}'$ differs from the construction presented in Theorem 4.2.1 only in the definition of the acceptance conditions. The assumption on the existence of uniform fin-acceptance synchronized runs is needed for showing that all words fin-accepted by $\mathcal{A}$ are included in the language fin-recognized by the automaton $\mathcal{A}'$ (which is obviously nondeterministic by definition).

($\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}')$) Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$. By the assumption, $\mathcal{A}$ has a uniform fin-acceptance synchronized run $G = \langle V, E, L \rangle$ on w. Let $G' = \langle V', E', L' \rangle$, V'_i (the levels of G' for all $0 \leq i < \omega$) and $T_i \subseteq \Delta$ (the transitions that label edges starting from nodes at level $0 \leq i < \omega$ of G,

$T_i = \bigcup_{1 \leq j \leq n_i} \{\langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j}\rangle\}$) be as defined in the corresponding direction in the proof of Theorem 4.2.1 with the exception that we define the label of the edge $\langle v'_i, V'_{i+1}\rangle \in E'$ $(0 \leq i < \omega)$ as $L'\big(\langle v'_i, V'_{i+1}\rangle\big) \stackrel{def}{=} \big\langle \bigcup_{1 \leq j \leq n_i}\{q_{i,j}\},\ \bigcap_{1 \leq j \leq n_i}\Gamma_{i,j},\ \bigcup_{1 \leq j \leq n_i} F_{i,j},\ \big\{\bigcup_{1 \leq j \leq n_i} Q'_{i,j}\big\}\big\rangle$ to make the edge labels match the transitions of $\mathcal{A}'$, which have their acceptance conditions chosen directly from the set $\mathcal{F}$. Similarly to Theorem 4.2.1, it is then straightforward to check that G' is a run of $\mathcal{A}'$ on w. Note that this check does not require assuming that $\mathcal{A}$ is a self-loop alternating automaton.

Suppose that G' is not fin-accepting. Then there exists an index $0 \leq j < \omega$ and an acceptance condition $f \in \mathcal{F}$ such that the transition $L'\big(\langle v'_i, V'_{i+1}\rangle\big)$ is an f-transition for all $j \leq i < \omega$, and thus T_i contains an f-transition for all $j \leq i < \omega$. But then G has only finitely many levels with no outgoing edges labeled with f-transitions. This contradicts the assumption that G is fin-acceptance synchronized (in particular, fin-synchronized with respect to f). Therefore G' is fin-accepting, and $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}')$ holds.

$(\mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}))$ Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}')$. It is again easy to check that the graph $G = \langle V, E, L\rangle$ defined in the corresponding direction of the proof of Theorem 4.2.1 (from a uniform fin-accepting run $G' = \langle V', E', L'\rangle$ of the nondeterministic automaton $\mathcal{A}'$ on w) is a run of $\mathcal{A}$ on w. (Because G' is a uniform run of a nondeterministic automaton, then $V'_i = \{v'_i\}$ is a singleton for all $0 \leq i < \omega$.)

If G is not fin-accepting, then there exists an infinite branch $(e_i)_{0 \leq i < \omega} \in \mathcal{B}(G)$, an acceptance condition $f \in \mathcal{F}$, and an index $0 \leq j < \omega$ such that the transition $L(e_i)$ is an f-transition for all $j \leq i < \omega$. Because $L(e_i) \in \Delta$ is (by definition of $\mathcal{A}'$) a "component" in the transition $L'\big(\big\langle v'_i, \{v'_{i+1}\}\big\rangle\big)$ that labels the edge between levels i and $i+1$ of G' (cf. the proof of Theorem 4.2.1), it follows that $L'\big(\big\langle v'_i, \{v'_{i+1}\}\big\rangle\big)$ is also an f-transition for all $j \leq i < \omega$. This is a contradiction, because G' was assumed to be fin-accepting. We conclude that $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds. $\square$

Alternating automata that have uniform fin-acceptance synchronized runs on all words in their language can thus be translated into nondeterministic automata using the universal subset construction without introducing new acceptance conditions. Instead, the acceptance conditions of a transition in the nondeterministic automaton can be defined simply as the union of the conditions in its component transitions. Because the translation is based on the universal subset construction, it is again easy to see that the nondeterministic automaton built from an alternating automaton with n states and m transitions has at most 2^n states and 2^m transitions.

4.3.3 Sufficient Conditions for Acceptance Synchronization

In this section we define a subclass of alternating automata which are guaranteed to have uniform fin-acceptance synchronized runs on all words in their language (and which can thus be translated into nondeterministic automata using the construction of Theorem 4.3.2). We first introduce the class and then show that all automata in the class have this property.

Terminology

We use the following terminology for characterizing our class of alternating automata. Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a fixed alternating automaton.

Let $\langle q, \Gamma, F, Q' \rangle \in \Delta$ be a transition of $\mathcal{A}$, and let $f \in \mathcal{F}$ be an acceptance condition. We say that the transition is *f-closed* iff $f \in F$ implies that Q' contains an f-state. The automaton $\mathcal{A}$ is *f-closed* iff all of its transitions are f-closed. Intuitively, this means that every finite chain of f-transitions in $\mathcal{A}$ can be extended into an infinite one. Finally, $\mathcal{A}$ is *acceptance closed (with respect to $\mathcal{F}$)* iff it is f-closed for all acceptance conditions $f \in \mathcal{F}$.

Example 4.3.3 The self-loop alternating automaton shown in Fig. 2.7 (p. 37) is not $\bullet$-closed, because it contains the transition $\big\langle q_6, \{b\}, \{\bullet\}, \{q_8\}\big\rangle$ having no $\bullet$-states as its target states. The automaton is $\circ$-closed, however. The automaton obtained from this automaton by removing acceptance conditions from its non-self-loop transitions (Fig. 2.8, p. 39) is both $\bullet$- and $\circ$-closed and therefore acceptance closed (with respect to the set of acceptance conditions $\{\bullet, \circ\}$). ∎

Let $q \in Q$. By Proposition 2.3.15, the subautomaton $\mathcal{A}^q$ fin-accepts a word $w \in \Sigma^\omega$ iff it has an initial transition $\langle q, \Gamma, F, Q' \rangle \in \Delta$ such that $w(0) \in \Gamma$ holds, and $(\mathcal{A}^q)^{q'}$ $(= \mathcal{A}^{q'})$ fin-accepts w^1 for all $q' \in Q'$. We say that $\mathcal{A}^q$ *fin-accepts w by avoiding an initial f-transition* (notation: $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$) if one of the transitions satisfying this condition is not an f-transition of $\mathcal{A}$. (Obviously, if q is not an f-state of $\mathcal{A}$, then $\mathcal{L}^f_{\text{fin}}(\mathcal{A}^q) = \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds.)

Let $f \in \mathcal{F}$ be an acceptance condition. We say that the state $q \in Q$ is an *f-representative state* iff q is an f-state, and for all $w \in \Sigma^\omega$,

- if $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ holds for some f-state $q' \in Q$, then $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q'})$, and
- if $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q'})$ holds for some f-state $q' \in Q$, then there exists an index $0 \leq i < \omega$ such that $w^i \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$ holds.

Intuitively, if the alternating automaton $\mathcal{A}$ has an f-representative state for one of its acceptance conditions $f \in \mathcal{F}$, then the f-states that occur as labels of nodes in a fin-accepting run of $\mathcal{A}$ determine certain input positions at which the active copies of the automaton could (but do not have to) "cooperate" with respect to the condition f by not taking f-transitions. If a level in the run contains a node labeled with an f-representative state such that the edge starting from this node is not labeled with an f-transition, then all active subautomata of the automaton could fin-accept the input from that position onward by avoiding an initial f-transition. On the other hand, if the run contains a node labeled with another (not necessarily f-representative) f-state such that the edge starting from this node is not labeled with an f-transition, then there exists a subsequent input position at which all active copies of the automaton could avoid taking f-transitions. As we shall show later in this section, the existence of an f-representative state in an f-closed alternating automaton implies that every fin-accepting run of the automaton can be synchronized with respect to the condition f.

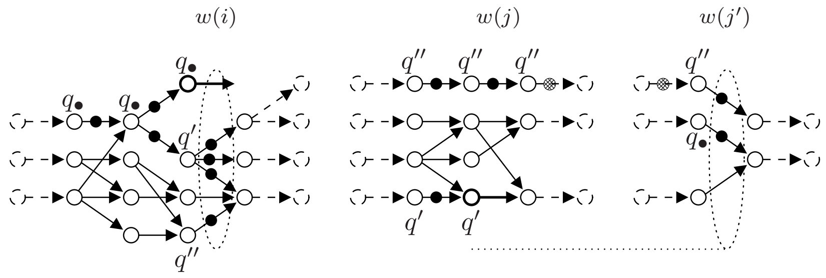


Fig. 4.4: Implications of the occurrence of $\bullet$-states as node labels in a fin-accepting run of an alternating automaton

Example 4.3.4 Figure 4.4 depicts parts of a fin-accepting run of an alternating automaton $\mathcal{A}$ with a $\bullet$-representative state $q_\bullet$ on an input w. The states q' and q'' are two nonrepresentative $\bullet$-states of the automaton; the circled levels identify input positions at which copies of the automaton in any $\bullet$-state could (but do not need to) avoid taking a $\bullet$-transition. The justification for this is given by the transition corresponding to the rightmost thick edge that starts from the same or a preceding level of the run. Formally, because $w^i \in \mathcal{L}^\bullet_{\text{fin}}(\mathcal{A}^{q_\bullet}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q''})$ and $w^j \in \mathcal{L}^\bullet_{\text{fin}}(\mathcal{A}^{q'})$ hold, it follows from the definition of representative states that $w^i \in \mathcal{L}^\bullet_{\text{fin}}(\mathcal{A}^{q'}) \cap \mathcal{L}^\bullet_{\text{fin}}(\mathcal{A}^{q''})$ and $w^{j'} \in \mathcal{L}^\bullet_{\text{fin}}(\mathcal{A}^{q_\bullet})$ hold (in which case also $w^{j'} \in \mathcal{L}^\bullet_{\text{fin}}(\mathcal{A}^{q''})$ holds). ■

Guaranteeing the Existence of Acceptance Synchronized Runs
Our main result in this section is the following:

Proposition 4.3.5 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an acceptance closed alternating automaton such that for all acceptance conditions $f \in \mathcal{F}$, if $\mathcal{A}$ has an f-state, then it has also an f-representative state. For all $w \in \Sigma^\omega$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds iff $\mathcal{A}$ has a uniform fin-acceptance synchronized run on w.*

To prove the proposition, we need one additional definition and a related result. Let $f \in \mathcal{F}$ be an acceptance condition of an f-closed alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ with an f-representative state $q_f \in Q$. Let $I_{\mathcal{A},f} : \Sigma^\omega \to 2^{\mathbb{N}}$ be a function defined, for all $w \in \Sigma^\omega$, by the rule

$$I_{\mathcal{A},f}(w) \overset{\text{def}}{=} \{0 \le i < \omega \,|\, w^i \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f})\}.$$

Intuitively, the function $I_{\mathcal{A},f}$ maps every word $w \in \Sigma^\omega$ to the set of indices i that identify the exact set of suffixes of the word w that are fin-recognized by the subautomaton $\mathcal{A}^{q_f}$ by avoiding an initial f-transition. This function is obviously well-defined because one of $w^i \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f})$ or $w^i \notin \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f})$ always holds for all $w \in \Sigma^\omega$ and $0 \le i < \omega$ (however, it need not be computable for all w in a finite number of steps). Clearly, if $|I_{\mathcal{A},f}(w)| < \omega$ holds, then the set $\{0\} \cup I_{\mathcal{A},f}(w)$ contains a maximal element. In this case it also follows that every subautomaton rooted at an f-state of $\mathcal{A}$ fin-accepts only finitely many suffixes of w:

Lemma 4.3.6 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an f-closed alternating automaton with an f-representative state $q_f \in Q$ for some $f \in \mathcal{F}$, and let $w \in \Sigma^\omega$.*

If $|I_{\mathcal{A},f}(w)| < \omega$ *holds, then* $w^i \notin \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ *holds for all* f*-states* $q \in Q$ *and indices* $\max\big(\{0\} \cup I_{\mathcal{A},f}(w)\big) < i < \omega$.

Proof: Because $I_{\mathcal{A},f}(w)$ is finite, $k \stackrel{def}{=} \max\big(\{0\} \cup I_{\mathcal{A},f}(w)\big) < \omega$ exists. Suppose that the subautomaton $\mathcal{A}^q$ fin-accepts w^i for some f-state $q \in Q$ and $k < i < \omega$, and let $G = \langle V, E, L\rangle$ be a fin-accepting run of $\mathcal{A}^q$ on w^i.

Let $\langle q, \Gamma, F, Q'\rangle \in \Delta^q$ ($\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$) be the initial transition of $\mathcal{A}^q$ labeling the edge between the first two levels of G. If $f \notin F$ holds, then $w^i \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$ holds. Because q_f is an f-representative state of $\mathcal{A}$, there exists an index $0 \leq j < \omega$ such that the subautomaton $\mathcal{A}^{q_f}$ fin-accepts $(w^i)^j = w^{i+j}$ by avoiding an initial f-transition. But then $i + j \in I_{\mathcal{A},f}(w)$ holds, which contradicts the assumption that $i > k = \max\big(\{0\} \cup I_{\mathcal{A},f}(w)\big)$ holds. Therefore the transition $\langle q, \Gamma, F, Q'\rangle$ is an f-transition.

Because $\mathcal{A}$ is f-closed, Q' contains an f-state $q' \in Q'$. Because L is consistent, the node v_0 at level 0 of G has a successor $v_1 \in V_1$ labeled with q'; furthermore, the subgraph of G rooted at v_1 is a fin-accepting run of $(\mathcal{A}^q)^{q'}$ ($= \mathcal{A}^{q'}$) on $(w^i)^1$ by Proposition 2.3.9. Because q' is an f-state of $\mathcal{A}$, a similar reasoning shows that also the edge starting from v_1 is labeled with an f-transition of $\mathcal{A}$, and v_1 has a successor labeled with an f-state of $\mathcal{A}$. By induction, it follows that G contains an infinite branch, all edges of which are labeled with f-transitions. But then $\mathcal{B}(G)$ contains an infinite branch that is not fin-accepting, and thus G cannot be fin-accepting, either, contrary to our assumption. Therefore $w^i \notin \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds. □

Proof of Proposition 4.3.5 (Only if): Without loss of generality, we may assume that $\mathcal{A}$ has an f-transition (hence, an f-state) for all acceptance conditions $f \in \mathcal{F}$: obviously, acceptance conditions not occurring in any transition of $\mathcal{A}$ can be discarded without affecting fin-acceptance. Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$. If $\mathcal{A}$ has no acceptance conditions ($\mathcal{F} = \emptyset$), then $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds iff $\mathcal{A}$ has a run on w, iff $\mathcal{A}$ has a uniform run on w (Proposition 4.1.1), and every run of $\mathcal{A}$ is trivially fin-acceptance synchronized. We may thus assume that $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ holds for some $1 \leq n < \omega$ such that $\mathcal{A}$ has an f_i-representative state $q_{f_i} \in Q$ for all $1 \leq i \leq n$. We define a uniform fin-acceptance synchronized run $G = \langle V, E, L\rangle$ of $\mathcal{A}$ on w.

(Definition of G) We define the levels of G inductively: first, let $V_0 \stackrel{def}{=} \{v_{0,1}\}$ and $L(v_{0,1}) \stackrel{def}{=} q_I$. For each level V_i in G ($0 \leq i < \omega$), we define also a set of edges E_i starting from nodes in V_i, and an integer $0 \leq c_i \leq n$ that is used to guide the inductive construction. To guarantee that G is fin-acceptance synchronized, we need to ensure that G is fin-synchronized with respect to all acceptance conditions $f \in \mathcal{F}$. The integers c_i are used to make sure that each acceptance condition is treated fairly in the construction: for all i, c_i will either have the special value 0, or it identifies an acceptance condition $f_{c_i} \in \mathcal{F}$ for which we should try to define a level having no outgoing edges labeled with f_{c_i}-transitions. Let $c_0 \stackrel{def}{=} 0$.

Assume that $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}$, $L(V_i) = \{q_{i,1}, q_{i,2}, \ldots, q_{i,n_i}\}$ (where $0 \leq n_i < \omega$, $L(v_{i,j}) = q_{i,j}$ and $L(v_{i,j}) \neq L(v_{i,k})$ hold for all $1 \leq j, k \leq n_i$, $j \neq k$), and c_i have already been defined for some $0 \leq i < \omega$, and assume also that $w^i \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_{i,j}})$ holds for all $1 \leq j \leq n_i$. (This is clear if $i = 0$, because $L(V_0) = \{q_I\}$, and $w^0 = w \in \mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$.)

Because $\mathcal{A}^{q_{i,j}}$ fin-accepts w^i for all $1 \le j \le n_i$, there exist transitions $t_{i,j} = \langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j} \rangle \in \Delta^{q_{i,j}} \subseteq \Delta$ (for some $\Gamma_{i,j} \subseteq \Sigma$, $F_{i,j} \subseteq \mathcal{F}$ and $Q'_{i,j} \subseteq Q^{q_{i,j}} \subseteq Q$) such that $w(i) \in \Gamma_{i,j}$ holds for all $1 \le j \le n_i$, and $\mathcal{A}^{q'}$ fin-accepts w^{i+1} for all $q' \in \bigcup_{1 \le j \le n_i} Q'_{i,j}$ (Proposition 2.3.15). Furthermore, if $c_i \neq 0$, and $q_{i,j}$ is not an f_{c_i}-state or if $i \in I_{\mathcal{A},f_{c_i}}(w)$ holds (in which case $w^i \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_{f_{c_i}}})$ holds for the f_{c_i}-representative state $q_{f_{c_i}}$), it follows that $\mathcal{A}^{q_{i,j}}$ fin-accepts w^i by avoiding an initial f-transition, and thus the transition $t_{i,j}$ can be chosen so that $f_{c_i} \notin F_{i,j}$ holds. In other words, if $i \in I_{\mathcal{A},f_{c_i}}(w)$ holds, then we can choose the transitions $t_{i,j}$ such that $f_{c_i} \notin \bigcup_{1 \le j \le n_i} F_{i,j}$ holds.

Let $\bigcup_{1 \le j \le n_i} Q'_{i,j} = \{q_{i+1,1}, q_{i+1,2}, \ldots, q_{i+1,n_{i+1}}\}$ (where $0 \le n_{i+1} < \omega$, and the states $q_{i+1,j}$ are pairwise distinct). Define V_{i+1} as a set of n_{i+1} new nodes $V_{i+1} \overset{\text{def}}{=} \{v_{i+1,1}, v_{i+1,2}, \ldots, v_{i+1,n_{i+1}}\}$, and let $L(v_{i+1,j}) \overset{\text{def}}{=} q_{i+1,j}$ for all $1 \le j \le n_{i+1}$. By construction, no two nodes in V_{i+1} have the same label, and because $L(V_{i+1}) = \bigcup_{1 \le j \le n_i} Q'_{i,j}$, it follows that for every $1 \le j \le n_i$, there exists a unique subset $V'_{i,j} \subseteq V_{i+1}$ such that $L(V'_{i,j}) = Q'_{i,j}$ holds. Let $E_i \overset{\text{def}}{=} \big\{\langle v_{i,j}, V'_{i,j} \rangle \,\big|\, 1 \le j \le n_i\big\}$, and for all $1 \le j \le n_i$, let $L\big(\langle v_{i,j}, V'_{i,j}\rangle\big) \overset{\text{def}}{=} t_{i,j}$. Finally, let

$$c_{i+1} \overset{\text{def}}{=} \begin{cases} c_i & \text{if } c_i \neq 0,\ |I_{\mathcal{A},f_{c_i}}(w)| = \omega \text{ and } \\ & \qquad i \notin I_{\mathcal{A},f_{c_i}}(w) \\ (c_i + 1) \bmod \big(|\mathcal{F}| + 1\big) & \text{otherwise.} \end{cases}$$

This completes the inductive definition of V_{i+1}, E_i and c_{i+1}; clearly, the construction ensures also that $w^{i+1} \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{L(v)})$ holds for all $v \in V_{i+1}$. We then define $V \overset{\text{def}}{=} \bigcup_{0 \le i < \omega} V_i$ and $E \overset{\text{def}}{=} \bigcup_{0 \le i < \omega} E_i$.

(G is a uniform run of $\mathcal{A}$ on w) It is clear that each level of G consists of nodes with distinct labels, and thus G satisfies the constraint required of uniform runs of $\mathcal{A}$. We check that G is a run of $\mathcal{A}$ on w.

(Partitioning) Obviously $V_0 = \{v_{0,1}\}$ is a singleton, and E consists of edges connecting pairwise disjoint consecutive levels of G by definition.

(Forward causality) Let $v \in V_i$ for some $0 \le i < \omega$. Then $v = v_{i,j}$ holds for some $1 \le j \le n_i$. Clearly, $E_i \subseteq E$ contains the edge $\langle v_{i,j}, V'_{i,j}\rangle$, and this is the only edge starting from $v_{i,j}$ in G.

(Backward causality) Let $v' \in V_i$ for some $1 \le i < \omega$. Then there exists a transition $t = \langle q, \Gamma, F, Q' \rangle$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$ such that $L(v') \in Q'$ holds, and V_{i-1} contains a node v with $L(v) = q$. By the definition of E, the node v has the outgoing edge $\langle v, V' \rangle \in E$ such that $L(V') = Q'$ holds. Because no two nodes at level V_i have the same label, it follows that $v' \in V'$ holds, and thus v' is a successor of the node v at level V_{i-1} in G.

(Consistency of L) Clearly $L(v_{0,1}) = q_I$. Let $e = \langle v_{i,j}, V'_{i,j}\rangle \in E$ be an edge in E for some $0 \le i < \omega$ and $1 \le j \le n_i$. By the definition of L, this edge is labeled with the transition $t_{i,j} = \langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j}\rangle \in \Delta$ such that $w(i) \in \Gamma_{i,j}$ holds. The definition also guarantees that $q_{i,j} = L(v_{i,j})$ and $Q'_{i,j} = L(V'_{i,j})$ hold, and thus the labeling L is consistent.

(G is fin-acceptance synchronized) We show that G is fin-acceptance synchronized. First, it is easy to see that $c_{i+1} \neq c_i$ holds for infinitely many

indices $0 \leq i < \omega$: otherwise there would exist an index $0 \leq i < \omega$ such that $c_j = c_i$ and $f_{c_j} = f_{c_i} = f$ hold for all $i \leq j < \omega$. By the definition of the integers c_j, it follows that $c_j = c_i \neq 0$, $|I_{\mathcal{A},f}(w)| = \omega$ and $j \notin I_{\mathcal{A},f}(w)$ hold for all $i \leq j < \omega$. But this is impossible, because the assumption that $j \notin I_{\mathcal{A},f}(w)$ holds for all $i \leq j < \omega$ would imply that $|I_{\mathcal{A},f}(w)| < \omega$ holds, a contradiction. Therefore, $c_{i+1} \neq c_i$ holds for infinitely many i, and because the integers c_i are defined incrementally modulo $|\mathcal{F}| + 1$, it follows that for all $f \in \mathcal{F}$, there are infinitely many indices $0 \leq i < \omega$ such that $c_i \neq 0$ and $f_{c_i} = f$ hold.

Let $f \in \mathcal{F}$. If $|I_{\mathcal{A},f}(w)| < \omega$ holds, then $k \overset{def}{=} \max\big(\{0\} \cup I_{\mathcal{A},f}(w)\big) < \omega$ exists. Suppose that E_i contains an edge labeled with an f-transition of $\mathcal{A}$ for some $k < i < \omega$. Because the labeling L is consistent, the source node $v \in V_i$ of this edge is labeled with an f-state of $\mathcal{A}$, and $\mathcal{A}^{L(v)}$ fin-accepts w^i by the inductive definition of G. However, this is impossible by Lemma 4.3.6. Therefore G contains only finitely many levels with an outgoing edge labeled with an f-transition of $\mathcal{A}$, and thus G is fin-synchronized with respect to the acceptance condition f.

Otherwise $|I_{\mathcal{A},f}(w)| = \omega$ holds. Let $0 \leq i < \omega$ be any of the (infinitely many) indices such that $c_i \neq 0$ and $f_{c_i} = f$ hold, and let $i \leq j < \omega$ be the least index greater than or equal to i for which $c_{j+1} \neq c_i$ holds. By the definition of c_{j+1}, it follows that $j \in I_{\mathcal{A},f}(w)$ holds in this case. But then the definition of G guarantees that E_j has no edges labeled with an f-transition of $\mathcal{A}$. It follows that G is fin-synchronized with respect to the acceptance condition f.

Because f is arbitrary, G is fin-synchronized with respect to all of its acceptance conditions, and thus G is a uniform fin-acceptance synchronized run of $\mathcal{A}$ on w.

(If) This direction follows immediately from Proposition 4.3.1. □

The following result is an immediate consequence of Proposition 4.3.5.

Corollary 4.3.7 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an acceptance closed alternating automaton that has an f-representative state for all acceptance conditions in $\mathcal{F}$ for which it has an f-state. The automaton $\mathcal{A}$ can be translated into a fin-equivalent nondeterministic automaton without introducing new acceptance conditions by applying the construction presented in Theorem 4.3.2.*

Proof: By Proposition 4.3.5, $\mathcal{A}$ fin-accepts a word $w \in \Sigma^\omega$ iff $\mathcal{A}$ has a uniform fin-acceptance synchronized run on w. Obviously, this is exactly the precondition for applying Theorem 4.3.2. □

4.3.4 Application to Translation of LTL into Nondeterministic Automata

As a simple first example on using the optimized nondeterminization construction of Theorem 4.3.2, we consider automata built from LTL formulas using the translation rules presented in Sect. 3.1. (The results of this section are needed further in Ch. 5 where we consider alternative translation rules.) It is easy to see that all automata built using the rules are acceptance closed alternating automata that have representative states for all of their acceptance conditions.

Lemma 4.3.8 *Let $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton built from an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ using the translation rules presented in Sect. 3.1. The automaton $\mathcal{A}$ is an acceptance closed automaton that has an f-representative state for every acceptance condition $f \in \mathcal{F}$.*

Proof: Let $f \in \mathcal{F}$ be an acceptance condition. As observed in Sect. 3.1.1, all f-transitions of $\mathcal{A}$ are self-loops of $\mathcal{A}$, and they have the same source state $q_f \in Q$ (which corresponds to the initial state of a subautomaton built for a binary pure temporal subformula of φ with a strong binary temporal main connective). Therefore $\mathcal{A}$ is f-closed, and the state q_f is trivially an f-representative state, because it is the only f-state of $\mathcal{A}$. The result follows because the same holds for all acceptance conditions $f \in \mathcal{F}$. $\square$

Consequently, the following result holds for the sizes of components in a nondeterministic automaton built from an LTL formula.

Corollary 4.3.9 *Let $\varphi \in LTL(AP)$ be an LTL formula. The language $\mathcal{L}(\varphi)$ can be fin-recognized by a nondeterministic automaton with at most $1 + 2^{|\mathrm{Temp}([\varphi]^{\mathrm{PNF}})|}$ states (at most $2^{|\mathrm{Temp}([\varphi]^{\mathrm{PNF}})|}$ states if φ is a binary pure temporal formula), $2^{O(|\varphi|)}$ transitions, and at most $n \stackrel{def}{=} \big|\{(\varphi_1 \circ \varphi_2) \in \mathrm{Sub}\big([\varphi]^{\mathrm{PNF}}\big) : \circ \in \{\mathsf{U_s}, \mathsf{R_s}\}\}\big|$ acceptance conditions.*

Proof: Let $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F} \rangle$ be the self-loop alternating automaton built from $[\varphi]^{\mathrm{PNF}}$ using the translation rules presented in Sect. 3.1. It is clear from the discussion in Sect. 3.2.3 that this automaton has at most n acceptance conditions. By Lemma 4.3.8, $\mathcal{A}$ is an acceptance closed automaton with representative states for all of its acceptance conditions, and it follows by Corollary 4.3.7 that $\mathcal{A}$ can be translated into a nondeterministic automaton with at most n acceptance conditions using the construction of Theorem 4.3.2. Because this construction is based on the universal subset construction, the upper bound for the number of states follows from the discussion in Sect. 4.2.2.

By the definition of the construction in Theorem 4.3.2, every transition in the nondeterministic automaton is an element of the set $2^Q \times 2^{2^{AP}} \times 2^{\mathcal{F}} \times 2^{2^Q}$. Because the target state set of every transition consists of only a single subset of Q, however, it is easy to see that the size of the set of possible target state sets is bounded by $2^{|Q|}$. Additionally, because the guards of transitions in the alternating automaton $\mathcal{A}$ are conjunctions of atomic formulas (Boolean constants or literals referring to atomic propositions that occur in the formula $[\varphi]^{\mathrm{PNF}}$), also the guards in the nondeterministic automaton can be written as such conjunctions. Furthermore, it is easy to see from the semantics of LTL that no Boolean constant or a literal need occur in any conjunction twice; therefore, because the order of the atomic formulas in a conjunction is not relevant, either, the number of these conjunctions is bounded by $2^{2\cdot|\varphi|}$. Because also $|Q| \in O\big(|\varphi|\big)$ and $|\mathcal{F}| = n \in O\big(|\varphi|\big)$ hold, it follows that the nondeterministic automaton has $2^{O(|\varphi|)}$ transitions as argued. $\square$

Admittedly, the above result is almost embarrassingly trivial at this point in light of the number of theoretical definitions and results we used to reach it. As a matter of fact, we could have obtained this result by appealing directly to the properties of the rules for translating LTL formulas into alternating automata: because the self-loop alternating automaton $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ built from the positive normal form of an LTL formula $\varphi \in LTL(AP)$ has only one f-state for every acceptance condition $f \in \mathcal{F}$ (see Sect. 3.1.1), it is easy to conclude by the discussion in Sect. 4.2.3 that no more than $|\mathcal{F}|$ acceptance conditions are needed for the nondeterministic automaton built from $\mathcal{A}$, even if it were built using the general nondeterminization construction of Theorem 4.2.1. The results presented in this section will prove to be useful in the next chapter, where we introduce new translation rules which may sometimes increase the number of f-states (for an acceptance condition f) in the self-loop alternating automaton. By showing that the new rules preserve acceptance closure and the existence of representative states for the acceptance conditions, we can then appeal to Corollary 4.3.7 to conclude that the automaton can still be translated into a nondeterministic automaton by using the universal subset construction of Theorem 4.3.2.

As a further theoretical curiosity, we note (analogously to Hammer et al. [2005]) that acceptance closed self-loop alternating automata with representative states for their acceptance conditions provide a "normal form" for self-loop alternating automata. A naive effective procedure for translating any self-loop alternating automaton into this form can be obtained by first translating the automaton into a linear temporal logic formula (Sect. 3.4) and then translating this formula back into an alternating automaton using the translation rules of Sect. 3.1; this automaton is in the normal form by Lemma 4.3.8. If the original automaton has n states and m acceptance conditions, then, by Proposition 3.4.4 and Corollary 3.2.2, the automaton obtained via this construction has $O\big(n(1+m)\big)$ states and as many acceptance conditions.

4.4 LANGUAGES ACCEPTED BY SUBAUTOMATA OF A NONDETERMINISTIC AUTOMATON

We note here a simple consequence of Theorem 4.2.1 and Theorem 4.3.2 that extends them into subautomata of the nondeterministic automaton obtained by applying the construction of Theorem 4.2.1 to a self-loop alternating automaton, or the construction of Theorem 4.3.2 to an acceptance closed alternating automaton that has representative states for all of its acceptance conditions.

Proposition 4.4.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ be a self-loop alternating automaton, or an acceptance closed alternating automaton with an f-representative state for all acceptance conditions $f \in \mathcal{F}$ for which it has an f-state, and let $\mathcal{A}_{\mathrm{nd}} = \big\langle \Sigma, 2^Q, \Delta_{\mathrm{nd}}, \{q_I\}, \mathcal{F}_{\mathrm{nd}}\big\rangle$ be the nondeterministic automaton obtained from $\mathcal{A}$ using the construction of Theorem 4.2.1 or Theorem 4.3.2, respectively. For all $Q' \subseteq Q$, $\bigcap_{q\in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \mathcal{L}_{\mathrm{fin}}\big((\mathcal{A}_{\mathrm{nd}})^{Q'}\big)$ holds.*

<u>*Proof:*</u> The result obviously holds if $\Sigma = \emptyset$ (in this case $\bigcap_{q\in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \mathcal{L}_{\mathrm{fin}}\big((\mathcal{A}_{\mathrm{nd}})^{Q'}\big) = \emptyset$). Assume that $\Sigma \neq \emptyset$ holds. Let $\sigma \in \Sigma$ be a fixed symbol of the alphabet Σ.

Let $\mathcal{A}^+ \overset{def}{=} \langle \Sigma, Q^+, \Delta^+, q_I^+, \mathcal{F}\rangle$ be the alternating automaton obtained from $\mathcal{A}$ by defining $Q^+ \overset{def}{=} Q \cup \{q_I^+\}$ for some new state q_I^+ not included in Q, and $\Delta^+ \stackrel{def}{=} \Delta \cup \big\{\langle q_I^+, \{\sigma\}, \emptyset, Q'\rangle\big\}$. Because no transition of $\mathcal{A}^+$ includes the state q_I^+ in its target states, $\mathcal{A}^+$ has the same loops as $\mathcal{A}$, and thus $\mathcal{A}^+$ is a self-loop alternating automaton if $\mathcal{A}$ is such an automaton. For the same reason, it is easy to see that $(\mathcal{A}^+)^q = \mathcal{A}^q$ holds for all $q \in Q$. Furthermore, because q_I^+ is not an f-state for any $f \in \mathcal{F}$, $\mathcal{A}^+$ has the same f-states, f-transitions, and subautomata rooted at an f-state as $\mathcal{A}$ for all acceptance conditions $f \in \mathcal{F}$. Therefore, if $\mathcal{A}$ is an acceptance closed alternating automaton with f-representative states for all $f \in \mathcal{F}$ for which it has an f-state, then so is $\mathcal{A}^+$.

It follows that the automaton $\mathcal{A}^+$ can be translated into a fin-equivalent nondeterministic automaton $\mathcal{A}^+_{\mathrm{nd}} = \big\langle \Sigma, 2^{Q^+}, \Delta^+_{\mathrm{nd}}, \{q_I^+\}, \mathcal{F}^+_{\mathrm{nd}}\big\rangle$ by applying the same construction that was used to build the nondeterministic automaton $\mathcal{A}_{\mathrm{nd}}$ from $\mathcal{A}$. It is easy to see that $\mathcal{A}^+_{\mathrm{nd}}$ has the unique initial transition $\big\langle \{q_I^+\}, \{\sigma\}, \emptyset, \{Q'\}\big\rangle \in \Delta^+_{\mathrm{nd}}$, and the target state of every transition of $\mathcal{A}^+_{\mathrm{nd}}$ is a subset of Q ($\mathcal{A}^+$ has no transitions having q_I^+ as a target state). Furthermore, $(\mathcal{A}^+_{\mathrm{nd}})^{Q''} = (\mathcal{A}_{\mathrm{nd}})^{Q''}$ holds for all $Q'' \subseteq Q$.

Let $w \in \Sigma^\omega$. Now, $w \in \bigcap_{q\in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$ holds

iff $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$ holds for all $q \in Q'$ *(definition of set intersection)*

iff $\mathcal{A}^q$ fin-accepts w for all $q \in Q'$ *(definition of $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$)*

iff $(\mathcal{A}^+)^q$ fin-accepts w for all $q \in Q'$ *($(\mathcal{A}^+)^q = \mathcal{A}^q$ for all $q \in Q \supseteq Q'$)*

iff there exists a transition $\big\langle q_I^+, \Gamma, F, Q''\big\rangle \in \Delta^+$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q'' \subseteq Q^+$ such that $\sigma \in \Gamma$ holds, and $(\mathcal{A}^+)^q$ fin-accepts w for all $q \in Q''$ *($\mathcal{A}^+$ has the unique initial transition $\langle q_I^+, \{\sigma\}, \emptyset, Q'\rangle$)*

iff $\mathcal{A}^+$ fin-accepts σw *(Proposition 2.3.15)*

iff $\mathcal{A}^+_{\mathrm{nd}}$ fin-accepts σw *($\mathcal{L}_{\mathrm{fin}}(\mathcal{A}^+_{\mathrm{nd}}) = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^+)$)*

iff there exists a transition $\big\langle \{q_I^+\}, \Gamma, F, \mathcal{Q}\big\rangle \in \Delta^+_{\mathrm{nd}}$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}^+_{\mathrm{nd}}$ and $\mathcal{Q} \subseteq 2^{Q^+}$ such that $\sigma \in \Gamma$ holds, and $(\mathcal{A}^+_{\mathrm{nd}})^{Q''}$ fin-accepts w for all $Q'' \in \mathcal{Q}$ *(Proposition 2.3.15)*

iff $(\mathcal{A}^+_{\mathrm{nd}})^{Q'}$ fin-accepts w *($\mathcal{A}^+_{\mathrm{nd}}$ has the unique initial transition $\big\langle \{q_I^+\}, \{\sigma\}, \emptyset, \{Q'\}\big\rangle \in \Delta^+_{\mathrm{nd}}$)*

iff $(\mathcal{A}_{\mathrm{nd}})^{Q'}$ fin-accepts w *($(\mathcal{A}^+_{\mathrm{nd}})^{Q'} = (\mathcal{A}_{\mathrm{nd}})^{Q'}$)*

iff $w \in \mathcal{L}_{\mathrm{fin}}\big((\mathcal{A}_{\mathrm{nd}})^{Q'}\big)$ holds. *(definition of $\mathcal{L}_{\mathrm{fin}}\big((\mathcal{A}_{\mathrm{nd}})^{Q'}\big)$)* □

Proposition 4.4.1 reduces the problem of checking for the fin-emptiness of the intersection of languages fin-recognized by subautomata of a self-loop alternating automaton (with properties given in the proposition) to checking for the fin-emptiness of the language of a nondeterministic automaton which can be found by applying the universal subset construction of Theorem 4.2.1 or Theorem 4.3.2.

4.5 ON-THE-FLY OPTIMIZATIONS TO NONDETERMINIZATION

In practice, the standard way to apply the universal subset construction used in Theorem 4.2.1 and Theorem 4.3.2 to build a nondeterministic automaton that recognizes the language of an alternating automaton (or an intersection of languages of its subautomata, cf. Sect. 4.4) is to build only that part of the nondeterministic automaton, the state set of which includes the initial state of the automaton and is closed under the automaton's transition relation. This phase is usually combined with on-the-fly heuristics to avoid generating some transitions [Gastin and Oddoux 2001; Fritz 2005] and to merge states with identical sets of outgoing transitions [Gastin and Oddoux 2001]; according to Gastin and Oddoux [2001], even such simple optimizations work well in practice to simplify the nondeterministic automaton. As formal evidence of this fact, Fritz [2005] showed a heuristic similar to the one suggested by Gastin and Oddoux [2001] for avoiding the generation of transitions in the nondeterministic automaton to cover comparable optimizations that are implicit in the tableau-based algorithm of Daniele et al. [1999] for translating LTL directly into nondeterministic automata.

Formally, Gastin and Oddoux's nondeterminization construction for very weak alternating automata implicitly uses the following result to avoid generating some transitions in the construction. We state this result without proof only for completeness.

Proposition 4.5.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton, and let $\mathcal{A}' = \langle \Sigma, 2^Q, \Delta', \{q_I\}, \mathcal{F}' \rangle$ be the nondeterministic automaton obtained from $\mathcal{A}$ using one the universal subset constructions of Theorem 4.2.1 or Theorem 4.3.2. Assume that the nondeterministic automaton $\mathcal{A}'$ has two transitions $t_1 = \langle Q_1', \Gamma_1, F_1, \{Q_1''\} \rangle \in \Delta'$ and $t_2 = \langle Q_2', \Gamma_2, F_2, \{Q_2''\} \rangle \in \Delta'$ (for some $Q_1', Q_2', Q_1'', Q_2'' \subseteq Q$, $\Gamma_1, \Gamma_2 \subseteq \Sigma$ and $F_1, F_2 \subseteq \mathcal{F}'$) such that $\Gamma_2 \supseteq \Gamma_1$, $F_2 \subseteq F_1$ and $Q_2'' \subseteq Q_1''$ hold. The automaton $\mathcal{A}'$ is fin-equivalent to the automaton $\mathcal{A}'' \overset{def}{=} \langle \Sigma, 2^Q, \Delta' \setminus \{t_1\}, \{q_I\}, \mathcal{F}' \rangle$ obtained from $\mathcal{A}'$ by removing the transition t_1 from Δ'.*

Intuitively, Proposition 4.5.1 implies that a nondeterminization construction does not need to generate a transition t_1 from a set of "component" transitions starting from a given set of states of the alternating automaton if the combination of another set of "component" transitions starting from these states corresponds to a transition t_2 such that t_1 and t_2 satisfy the above constraints.

We shall not consider the removal of transitions further in this section; instead, we review syntactic techniques originally proposed for direct translation algorithms between LTL and nondeterministic automata [Gerth et al. 1995; Daniele et al. 1999; Giannakopoulou and Lerda 2002] to illustrate their applications to reducing the size of the state set of the nondeterministic automaton obtained via the universal subset construction from a self-loop alternating automaton built from an LTL formula in positive normal form. These applications follow from the correspondence between the states of the alternating automaton and the node subformulas of the formula; by Proposition 4.4.1, every state of a nondeterministic automaton obtained from the alternating automaton corresponds to the conjunction of a collection of such subformulas.

4.5.1 Detecting Redundant States Using Syntactic Implications

By Corollary 2.3.11, the language recognized by any automaton remains the same after removing from the automaton all states, the subautomata rooted at which recognize the empty language. Given a self-loop alternating automaton built from an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$, it follows directly from this observation that the universal subset construction can ignore states which correspond to unsatisfiable conjunctions of node subformulas of φ. Whether a conjunction of a finite set of LTL formulas $\Phi \subseteq LTL^{\mathrm{PNF}}(AP)$ is unsatisfiable can be estimated heuristically with simple syntactic checks. For example, the formula $\bigwedge_{\psi\in\Phi} \psi$ is easily seen to be unsatisfiable if one of the following conditions holds:

- $\bot \in \Phi$, or $[\neg\psi]^{\mathrm{PNF}} \in \Phi$ for some $\psi \in \Phi$;
- $\big\{[\neg\psi_1]^{\mathrm{PNF}}, [\neg\psi_2]^{\mathrm{PNF}}\big\} \subseteq \Phi$ for some $(\psi_1 \circ \psi_2) \in \Phi$ ($\circ \in \{\vee, \mathsf{U_s}, \mathsf{U_w}\}$);
- $[\neg\psi_1]^{\mathrm{PNF}} \in \Phi$ for some $(\psi_1 \wedge \psi_2) \in \Phi$; or
- $[\neg\psi_2]^{\mathrm{PNF}} \in \Phi$ for some $(\psi_1 \circ \psi_2) \in \Phi$ ($\circ \in \{\wedge, \mathsf{R_s}, \mathsf{R_w}\}$).

To increase the likelihood of detecting a conjunction of a set of formulas to be unsatisfiable using syntactic checks, Daniele et al. [1999] suggested first extending this set with formulas that follow by syntactic implication from the formulas in the set. Such formulas can again be found by using simple rules to add formulas to the set. Formally, given a finite collection $\Phi' \subseteq LTL^{\mathrm{PNF}}(AP)$ of LTL formulas in positive normal form, Daniele et al. [1999] use a function $\mathcal{SI} : 2^{\Phi'} \to 2^{\Phi' \cup \{\top\}}$ that maps every subset $\Phi \subseteq \Phi'$ to the minimal subset of $\Phi' \cup \{\top\}$ that is closed under the following rules:

- $\Phi \cup \{\top\} \subseteq \mathcal{SI}(\Phi)$.
- If $(\psi_1 \vee \psi_2) \in \Phi'$ and $\psi_1 \in \mathcal{SI}(\Phi)$ hold, then $(\psi_1 \vee \psi_2) \in \mathcal{SI}(\Phi)$.
- If $(\psi_1 \circ \psi_2) \in \Phi'$ and $\psi_2 \in \mathcal{SI}(\Phi)$ hold for some $\circ \in \{\vee, \mathsf{U_s}, \mathsf{U_w}\}$, then $(\psi_1 \circ \psi_2) \in \mathcal{SI}(\Phi)$.
- If $(\psi_1 \circ \psi_2) \in \Phi'$ and $\big\{\psi_1, \mathsf{X}(\psi_1 \circ \psi_2)\big\} \subseteq \mathcal{SI}(\Phi)$ hold for some $\circ \in \{\mathsf{U_s}, \mathsf{U_w}\}$ then $(\psi_1 \circ \psi_2) \in \mathcal{SI}(\Phi)$.
- If $(\psi_1 \circ \psi_2) \in \Phi'$ and $\{\psi_1, \psi_2\} \subseteq \mathcal{SI}(\Phi)$ hold for some $\circ \in \{\wedge, \mathsf{R_s}, \mathsf{R_w}\}$, then $(\psi_1 \circ \psi_2) \in \mathcal{SI}(\Phi)$.
- If $(\psi_1 \circ \psi_2) \in \Phi'$ and $\big\{\psi_2, \mathsf{X}(\psi_1 \circ \psi_2)\big\} \subseteq \mathcal{SI}(\Phi)$ hold for some $\circ \in \{\mathsf{R_s}, \mathsf{R_w}\}$ then $(\psi_1 \circ \psi_2) \in \mathcal{SI}(\Phi)$.

(Of course, the domain and range of $\mathcal{SI}$ need be restricted to finite sets of formulas only to ensure that the computation of $\mathcal{SI}(\Phi)$ will eventually terminate; every LTL formula has infinitely many logically equivalent LTL formulas that differ from each other syntactically.)

The above rules can be used to infer the satisfaction of compound subformulas of Φ' from the satisfaction of other formulas. If Φ' is closed under taking of node subformulas (i.e., if $\mathrm{NSub}(\psi) \subseteq \Phi'$ holds for all $\psi \in \Phi'$), syntactic implication can be extended also in the converse direction by adding the following closure rules that are (except for the first one) actually part of the standard procedure of computing "covers" of LTL formulas, a basic operation used in tableau-based decision procedures for LTL [Manna and Wolper 1982, 1984; Wolper 1985] and related algorithms for translating LTL directly into nondeterministic automata [Gerth et al. 1995; Daniele et al. 1999]:

- If $\{(\psi_1 \circ \psi_2), [\neg\psi_i]^{\mathrm{PNF}}\} \subseteq \mathcal{SI}(\Phi)$ holds for some $\circ \in \{\vee, \mathsf{U_s}, \mathsf{U_w}\}$ and $i \in \{1, 2\}$, then $\psi_{3-i} \in \mathcal{SI}(\Phi)$.

- If $(\psi_1 \wedge \psi_2) \in \mathcal{SI}(\Phi)$ holds, then $\psi_1, \psi_2 \in \mathcal{SI}(\Phi)$.

- If $(\psi_1 \circ \psi_2) \in \mathcal{SI}(\Phi)$ holds for some $\circ \in \{\mathsf{R_s}, \mathsf{R_w}\}$, then $\psi_2 \in \mathcal{SI}(\Phi)$.

It is straightforward to check (by induction on the number of closure rules applied to a set of formulas Φ) from the semantics of LTL that $\bigwedge_{\psi\in\Phi} \psi \equiv \bigwedge_{\psi\in\mathcal{SI}(\Phi)} \psi$ holds. Therefore, if the conjunction of the formulas in $\mathcal{SI}(\Phi)$ is unsatisfiable, then so is the conjunction of formulas in Φ. In our application of translating an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ into a nondeterministic automaton via a self-loop alternating automaton, it is natural to use the set of φ's node subformulas ($\mathrm{NSub}(\varphi)$) as the set Φ'.

4.5.2 Merging Syntactically Language-Equivalent States

Let $\mathcal{SI}$ be the function (with a domain $2^{\Phi'}$ for some finite set of formulas $\Phi' \subseteq LTL^{\mathrm{PNF}}(AP)$) defined in the previous section. Clearly, if $\mathcal{SI}(\Phi_1) = \mathcal{SI}(\Phi_2)$ holds for two sets of LTL formulas $\Phi_1, \Phi_2 \subseteq \Phi'$, then also $\bigwedge_{\psi\in\Phi_1} \psi \equiv \bigwedge_{\psi\in\mathcal{SI}(\Phi_1)} \psi \equiv \bigwedge_{\psi\in\mathcal{SI}(\Phi_2)} \psi \equiv \bigwedge_{\psi\in\Phi_2} \psi$ holds. Therefore, if Φ_1 and Φ_2 correspond to two different states in a nondeterministic automaton built from an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ via a self-loop alternating automaton, it follows that the subautomata rooted at these states recognize the same language. The fact that the function $\mathcal{SI}$ thus induces an equivalence relation between states of the automaton hints at a possibility to reduce the number of states in the nondeterministic automaton by "merging" language-equivalent states (formally referred to as *quotienting* in the literature).

When working with automata on infinite words, language equivalence between two subautomata of a nondeterministic automaton is in general too weak a relation to guarantee that the automaton obtained by quotienting remains equivalent to the original automaton (see, for example, [Somenzi and Bloem 2000]). Therefore, algorithms for minimization by quotienting are usually based on equivalences under stronger *simulation relations* which guarantee the correctness of a quotient construction directly [Etessami and Holzmann 2000; Somenzi and Bloem 2000; Etessami et al. 2001, 2005; Etessami 2002], or admit checking whether it is safe to apply a quotient construction without resorting to a full language equivalence test between the original and the minimized automaton [Gurumurthy et al. 2002]. (Not all simulation equivalences can be safely used for quotienting, either—see, e.g., [Etessami et al. 2001, 2005; Bustan and Grumberg 2002, 2004] for discussion and examples.) We shall show that the function $\mathcal{SI}$ induces one of the simpler type of these equivalences: two states can be safely merged whenever they are found to be language-equivalent by computing syntactic implications. Instead of applying an explicit quotient construction to a nondeterministic automaton built using the universal subset construction of Theorem 4.3.2, we embed the construction directly into the definition of a nondeterministic automaton.

The idea of using equivalences based on syntactic implication to merge language-equivalent states in a nondeterministic automaton is well-known from tableau-based direct translation algorithms between LTL and nondeterministic automata [Daniele et al. 1999; Giannakopoulou and Lerda 2002]. The details of using syntactic implications to optimize nondeterminization constructions for alternating automata have not been considered in the literature, however. Our approach to minimization is nevertheless very similar to the more general approach of Fritz [2003], who used simulation relations on alternating automata [Fritz and Wilke 2002, 2005] (computed as an independent step without making use of correspondences between states and LTL formulas) for optimizing the nondeterminization construction of Miyano and Hayashi [1984a,b]. The simulation relations used in the approach are, however, targeted for alternating automata with state-based acceptance using a single acceptance condition (and an explicit partitioning of the states into "existential" and "universal" states [Miyano and Hayashi 1984a,b]) and are thus not directly compatible with our definition of alternating automata.

Optimizing Nondeterminization Using Syntactic Implications

In the following, let $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F} \rangle$ be a fixed alternating automaton built from a given LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ using the translation procedure outlined in Sect. 3.1. By the discussion in Sect. 3.1.1, there exists a bijective correspondence $\gamma : Q \to \mathrm{NSub}(\varphi)$ between the states of $\mathcal{A}$ and the node subformulas of φ such that (by the correctness of the translation procedure, Theorem 3.3.2) for all $q \in Q$, $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \mathcal{L}\big(\gamma(q)\big)$ holds. Let $\iota : 2^{\mathrm{NSub}(\varphi)} \to 2^{\mathrm{NSub}(\varphi)}$ be a function that maps every set of node subformulas of φ to another set of node subformulas of φ such that

$$\Phi \subseteq \iota(\Phi) \qquad \text{and} \qquad \mathcal{L}\Big(\bigwedge_{\psi \in \Phi} \psi\Big) \subseteq \mathcal{L}\Big(\bigwedge_{\psi \in \iota(\Phi)} \psi\Big)$$

hold for all subsets $\Phi \subseteq \mathrm{NSub}(\varphi)$. (Obviously, the first of these properties implies by the semantics of LTL that the second condition is in fact equivalent to the condition $\bigwedge_{\psi \in \Phi} \psi \equiv \bigwedge_{\psi \in \iota(\Phi)} \psi$.) For example, it is easy to see that the function $\mathcal{SI}$ defined in the previous section has these properties (technically, because $\top \in \mathcal{SI}(\Phi)$ always holds, we may need to replace the formula φ with e.g. the logically equivalent formula $(\varphi \wedge \top)$ to ensure that $\mathcal{SI}$ is closed with respect to $2^{\mathrm{NSub}(\varphi)}$). To simplify the notation, we treat the function ι as a mapping from 2^Q to 2^Q by writing (for a subset $Q' \subseteq Q$) $\iota(Q')$ in place of

$$\big\{\gamma^{-1}(\psi) \,\big|\, \psi \in \iota\big(\{\gamma(q) \mid q \in Q'\}\big)\big\}$$

(where γ^{-1} is the inverse of γ). In this notation, it follows from the properties of the functions ι and γ that

$$Q' \subseteq \iota(Q') \qquad \text{and} \qquad \bigcap_{q \in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \bigcap_{q \in \iota(Q')} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$$

hold for all subsets $Q' \subseteq Q$.

We intend to prove the following result:

Proposition 4.5.2 *Let $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F} \rangle$ and $\iota : 2^{\mathrm{NSub}(\varphi)} \to 2^{\mathrm{NSub}(\varphi)}$ be an alternating automaton (built from an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$) and a function (respectively) satisfying the above assumptions. Define the* *automaton* $\mathcal{A}_\iota = \big\langle 2^{AP}, \big\{\iota(Q') \bigm| Q' \subseteq Q\big\}, \Delta_\iota, \iota\big(\{q_I\}\big), \mathcal{F}\big\rangle$, *where, for all* $Q' \in \big\{\iota(Q'') \bigm| Q'' \subseteq Q\big\}$, $\Gamma \subseteq 2^{AP}$, $F \subseteq \mathcal{F}$ *and* $\mathcal{Q} \subseteq \big\{\iota(Q'') \bigm| Q'' \subseteq Q\big\}$,

$\big\langle Q', \Gamma, F, \mathcal{Q}\big\rangle \in \Delta_\iota$ *iff* *for all* $q \in Q'$, *there exists a transition* $\langle q, \Gamma_q, F_q, Q'_q\rangle \in \Delta$ *such that* $\Gamma = \bigcap_{q\in Q'} \Gamma_q$, $F = \bigcup_{q\in Q'} F_q$, *and* $\mathcal{Q} = \big\{\iota\big(\bigcup_{q\in Q'} Q'_q\big)\big\}$ *hold.*

The automaton $\mathcal{A}_\iota$ *is nondeterministic, and* $\mathcal{L}_{\text{fin}}(\mathcal{A}_\iota) = \mathcal{L}_{\text{fin}}(\mathcal{A})$ *holds.*

The definition of the automaton $\mathcal{A}_\iota$ is otherwise identical to the definition of a nondeterministic automaton in the universal subset construction of Theorem 4.3.2 except for changes used to ensure that the initial state and the transition relation of $\mathcal{A}_\iota$ refer to elements of the state set $\big\{\iota(Q') \bigm| Q' \subseteq Q\big\} \subseteq 2^Q$; it is easy to see that the automaton $\mathcal{A}_\iota$ is nondeterministic. Intuitively, instead of defining the target states of transitions of $\mathcal{A}_\iota$ as unions of the target states of their component transitions (from $\mathcal{A}$) as in the universal subset construction of Theorem 4.3.2, the construction in Proposition 4.5.2 uses the function ι to direct transitions to states that are equivalent under the relation induced by ι into a single representative state; also the initial state of the automaton is replaced with its representative. More precisely, the transitions of the automaton $\mathcal{A}_\iota$ are related in the following way to the transitions of the automaton defined from $\mathcal{A}$ using the construction in Theorem 4.3.2:

Lemma 4.5.3 *Let* $\mathcal{A}_\iota$ *and* $\mathcal{A}_{\text{nd}}$ *(with transitions* Δ_ι *and* Δ_{nd}*, respectively) be the nondeterministic automata obtained from the self-loop alternating automaton* $\mathcal{A}$ *using the constructions of Proposition 4.5.2 and Theorem 4.3.2, respectively. For all transitions* $\big\langle Q'_\iota, \Gamma_\iota, F_\iota, \{Q''_\iota\}\big\rangle \in \Delta_\iota$ *and all subsets* $Q' \subseteq Q'_\iota$, *the automaton* $\mathcal{A}_{\text{nd}}$ *has a transition* $\big\langle Q', \Gamma_{\text{nd}}, F_{\text{nd}}, \{Q''_{\text{nd}}\}\big\rangle \in \Delta_{\text{nd}}$ *such that* $\Gamma_{\text{nd}} \supseteq \Gamma_\iota$, $F_{\text{nd}} \subseteq F_\iota$ *and* $Q''_{\text{nd}} \subseteq Q''_\iota$ *hold.*

Proof: Let $\big\langle Q'_\iota, \Gamma_\iota, F_\iota, \{Q''_\iota\}\big\rangle \in \Delta_\iota$ be a transition of $\mathcal{A}_\iota$, and let $Q' \subseteq Q'_\iota$. By the definition of $\mathcal{A}_\iota$, there exists, for all $q \in Q'_\iota$, a transition $t_q = \langle q, \Gamma_q, F_q, Q'_q\rangle \in \Delta$ such that $\Gamma_\iota = \bigcap_{q\in Q'_\iota} \Gamma_q$, $F_\iota = \bigcup_{q\in Q'_\iota} F_q$ and $Q''_\iota = \iota\big(\bigcup_{q\in Q'_\iota} Q'_q\big)$ hold. Because $Q' \subseteq Q'_\iota$ holds, the collection of transitions $\{t_q \mid q \in Q'\}$ defines (in the automaton $\mathcal{A}_{\text{nd}}$) a transition $\big\langle Q', \Gamma_{\text{nd}}, F_{\text{nd}}, \{Q''_{\text{nd}}\}\big\rangle = \big\langle Q', \bigcap_{q\in Q'} \Gamma_q, \bigcup_{q\in Q'} F_q, \big\{\bigcup_{q\in Q'} Q'_q\big\}\big\rangle \in \Delta_{\text{nd}}$, and the result follows because

$$\begin{aligned}
\Gamma_{\text{nd}} &= \textstyle\bigcap_{q\in Q'} \Gamma_q \supseteq \big(\bigcap_{q\in Q'} \Gamma_q\big) \cap \big(\bigcap_{q\in Q'_\iota\setminus Q'} \Gamma_q\big) = \bigcap_{q\in Q'_\iota} \Gamma_q = \Gamma_\iota,\\
F_{\text{nd}} &= \textstyle\bigcup_{q\in Q'} F_q \subseteq \big(\bigcup_{q\in Q'} F_q\big) \cup \big(\bigcup_{q\in Q'_\iota\setminus Q'} F_q\big) = \bigcup_{q\in Q'_\iota} F_q = F_\iota, \text{ and}\\
Q''_{\text{nd}} &= \textstyle\bigcup_{q\in Q'} Q'_q \subseteq \big(\bigcup_{q\in Q'} Q'_q\big) \cup \big(\bigcup_{q\in Q'_\iota\setminus Q'} Q'_q\big) = \bigcup_{q\in Q'_\iota} Q'_q\\
&\subseteq \textstyle\iota\big(\bigcup_{q\in Q'_\iota} Q'_q\big) = Q''_\iota
\end{aligned}$$

hold. □

It is now easy to show that the automaton $\mathcal{A}_\iota$ defined in Proposition 4.5.2 from the alternating automaton $\mathcal{A}$ fin-recognizes no more words than $\mathcal{A}$.

Lemma 4.5.4 *Let* $\mathcal{A}$ *and* $\mathcal{A}_\iota$ *be the automata specified in Proposition 4.5.2. For all* $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_\iota)$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ *holds.*

Proof: Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_\iota)$. Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_{\text{nd}}) = \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds for the nondeterministic automaton $\mathcal{A}_{\text{nd}}$ (with transitions Δ_{nd}) obtained from the automaton $\mathcal{A}$ using the construction of Theorem 4.3.2, we prove the result by showing that $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_{\text{nd}})$ holds.

Let $G = \langle V, E, L\rangle$ be a fin-accepting run of $\mathcal{A}_\iota$ on w. Because $\mathcal{A}_\iota$ is a nondeterministic automaton, we may assume that G is a uniform run of $\mathcal{A}_\iota$ (Proposition 4.1.1). Therefore, V consists of an infinite number of levels V_i, each of which contains a single node $v_i \in V_i$ $(0 \le i < \omega)$, and G contains the unique infinite chain of edges $(e_i)_{0\le i<\omega} = \big(\langle v_i, \{v_{i+1}\}\rangle\big)_{0\le i<\omega}$. We define a relabeling L' of V and E to obtain a fin-accepting run $G' = \langle V, E, L'\rangle$ of $\mathcal{A}_{\text{nd}}$ on w. Let $L'(v_0) \stackrel{def}{=} \{q_I\}$.

Assume that $L'(v_i)$ has been defined for some $0 \le i < \omega$, and assume also that $L'(v_i) \subseteq L(v_i)$ holds (clearly, because G is a consistently labeled run of $\mathcal{A}_\iota$, this assumption holds for $i = 0$ because $L'(v_0) = \{q_I\} \subseteq \iota\big(\{q_I\}\big) = L(v_0)$ holds). Because G is a run of $\mathcal{A}_\iota$, the edge $e_i = \big\langle v_i, \{v_{i+1}\}\big\rangle$ is labeled with the transition $\big\langle L(v_i), \Gamma_i, F_i, \big\{L(v_{i+1})\big\}\big\rangle \in \Delta_\iota$ (for some $\Gamma_i \subseteq 2^{AP}$ and $F_i \subseteq \mathcal{F}$) such that $w(i) \in \Gamma_i$ holds. Because $L'(v_i) \subseteq L(v_i)$ holds, it follows by Lemma 4.5.3 that the automaton $\mathcal{A}_{\text{nd}}$ has a transition $t_i = \big\langle L'(v_i), \Gamma'_i, F'_i, \{Q''\}\big\rangle \in \Delta_{\text{nd}}$ for some $\Gamma'_i \supseteq \Gamma_i$, $F'_i \subseteq F_i$ and $Q'' \subseteq L(v_{i+1})$. We now define $L'(e_i) \stackrel{def}{=} t_i$ and $L'(v_{i+1}) \stackrel{def}{=} Q''$; clearly, $L'(v_{i+1}) \subseteq L(v_{i+1})$ holds, and we may complete the definition of the labeling L' by induction on i.

We claim that the graph $G' = \langle V, E, L'\rangle$ is a fin-accepting run of $\mathcal{A}_{\text{nd}}$ on w. Clearly, because G' shares its nodes and edges with the run G, G' satisfies the partitioning and causality constraints. Because $w(i) \in \Gamma_i \subseteq \Gamma'_i$ and $L'(e_i) = t_i = \big\langle L'(v_i), \Gamma'_i, F'_i, \big\{L'(v_{i+1})\big\}\big\rangle \in \Delta_{\text{nd}}$ hold for all $0 \le i < \omega$, it is easy to see that L' is consistent. Furthermore, because $\text{fin}\big((e_i)_{0\le i<\omega}\big) = \emptyset$ holds in G and $F'_i \subseteq F_i$ holds for all $0 \le i < \omega$, it follows that G' is a fin-accepting run of $\mathcal{A}_{\text{nd}}$ on w, and thus $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_{\text{nd}}) = \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds. $\square$

The following lemma establishes language containment in the converse direction.

Lemma 4.5.5 *Let $\mathcal{A}$ and $\mathcal{A}_\iota$ be the automata specified in Proposition 4.5.2. For all $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_\iota)$ holds.*

Proof: The proof relies on the fact that the automaton $\mathcal{A}$ is an acceptance closed alternating automaton that has an f-representative state for all of its acceptance conditions $f \in \mathcal{F}$ (Lemma 4.3.8). Using this result, a straightforward modification to the construction in the proof of Proposition 4.3.5 can be used to find a fin-acceptance synchronized run of $\mathcal{A}_\iota$ on w; thus $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_\iota)$ follows by Proposition 4.3.1. The details are as follows.

As in the proof of Proposition 4.3.5, write $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ for some $0 \le n < \omega$. By Lemma 4.3.8, $\mathcal{A}$ has an f_i-representative state $q_{f_i} \in Q$ for all $1 \le i \le n$. Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$; we define a fin-acceptance synchronized run $G = \langle V, E, L\rangle$ of $\mathcal{A}_\iota$ on w.

(Definition of G) Let $V \stackrel{def}{=} \bigcup_{0\le i<\omega} V_i = \bigcup_{0\le i<\omega}\{v_i\}$ (where $v_i \ne v_j$ holds for all $0 \le i, j < \omega$, $i \ne j$) and $E \stackrel{def}{=} \bigcup_{0\le i<\omega}\big\{\langle v_i, V_{i+1}\rangle\big\}$. Obviously, the

unlabeled graph $\langle V, E\rangle$ satisfies the partitioning and causality constraints. We define the labeling L inductively as follows.

Let $L(v_0) \overset{def}{=} \iota\bigl(\{q_I\}\bigr)$. Identically to the proof of Proposition 4.3.5, we define auxiliary integers $c_0, c_1, c_2, \ldots$ ($0 \le c_i \le n$ for all $0 \le i < \omega$) to guide the construction; let $c_0 \overset{def}{=} 0$.

Assume that $L(v_i)$ has been defined for some $0 \le i < \omega$; write $L(v_i) = \{q_{i,1}, q_{i,2}, \ldots, q_{i,n_i}\}$ for some $0 \le n_i < \omega$ such that $q_{i,j} \neq q_{i,k}$ holds for all $1 \le j, k \le n_i$ ($j \neq k$). Assume also that the labeling L is consistent up to level i of G (i.e., for all nodes $v_j \in V$, $0 \le j \le i$, and edges $\langle v_k, V_{k+1}\rangle \in E$, $0 \le k < i$), and that $w^i \in \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds for all $q \in L(v_i)$. These assumptions clearly hold if $i = 0$: $L(v_0) = \iota\bigl(\{q_I\}\bigr)$ is the initial state of $\mathcal{A}_\iota$, and because $w^0 = w \in \mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ holds (Proposition 2.3.12), it follows from the properties of the function ι and the definition of $L(v_0)$ that $w^0 \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I}) = \bigcap_{q\in\{q_I\}} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) = \bigcap_{q\in\iota(\{q_I\})} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) = \bigcap_{q\in L(v_0)} \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds, i.e., $\mathcal{A}^q$ fin-accepts w^0 for all $q \in L(v_0)$.

Because $\mathcal{A}^{q_{i,j}}$ fin-accepts w^i for all $1 \le j \le n_i$, there exist transitions $t_{i,j} = \langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j}\rangle \in \Delta^{q_{i,j}} \subseteq \Delta$ (for some $\Gamma_{i,j} \subseteq 2^{AP}$, $F_{i,j} \subseteq \mathcal{F}$ and $Q'_{i,j} \subseteq Q^{q_{i,j}} \subseteq Q$) such that $w(i) \in \Gamma_{i,j}$ holds for all $1 \le j \le n_i$, and $\mathcal{A}^{q'}$ fin-accepts w^{i+1} for all $q' \in \bigcup_{1\le j\le n_i} Q'_{i,j}$ (Proposition 2.3.15). Furthermore, if $c_i \neq 0$, and $q_{i,j}$ is not an f_{c_i}-state or if $i \in I_{\mathcal{A},f_{c_i}}(w)$ holds[3], it follows from the definition of f-representative states (Sect. 4.3.3) that $\mathcal{A}^{q_{i,j}}$ fin-accepts w^i by avoiding an initial f-transition, and thus the transitions $t_{i,j}$ can be chosen so that $f_{c_i} \notin \bigcup_{1\le j\le n_i} F_{i,j}$ holds.

We now extend the labeling L by defining

$$
\begin{aligned}
L(v_{i+1}) &\overset{def}{=} \iota\bigl(\textstyle\bigcup_{1\le j\le n_i} Q'_{i,j}\bigr),\\
L\bigl(\langle v_i, V_{i+1}\rangle\bigr) &\overset{def}{=} \bigl\langle \textstyle\bigcup_{1\le j\le n_i}\{q_{i,j}\}, \bigcap_{1\le j\le n_i}\Gamma_{i,j}, \bigcup_{1\le j\le n_i} F_{i,j},\\
&\qquad\qquad \bigl\{\iota\bigl(\textstyle\bigcup_{1\le j\le n_i} Q'_{i,j}\bigr)\bigr\}\bigr\rangle,
\end{aligned}
$$

and, as in the proof of Proposition 4.3.5,

$$
c_{i+1} \overset{def}{=} \begin{cases} c_i & \text{if } c_i \neq 0,\ |I_{\mathcal{A},f_{c_i}}(w)| = \omega \text{ and} \\ & \qquad\qquad\qquad i \notin I_{\mathcal{A},f_{c_i}}(w) \\ (c_i + 1) \bmod \bigl(|\mathcal{F}| + 1\bigr) & \text{otherwise.} \end{cases}
$$

Since $w^{i+1} \in \bigcap_{q'\in\bigcup_{1\le j\le n_i} Q'_{i,j}} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) = \bigcap_{q'\in\iota(\bigcup_{1\le j\le n_i} Q'_{i,j})} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ holds by the properties of the function ι, it follows that $\mathcal{A}^{q'}$ fin-accepts w^{i+1} for all $q' \in \iota\bigl(\bigcup_{1\le j\le n_i} Q'_{i,j}\bigr)$. Moreover, it is easy to see from the definition of $\mathcal{A}_\iota$ that $L\bigl(\langle v_i, V_{i+1}\rangle\bigr) \in \Delta_\iota$ holds, and because $w(i) \in \bigcap_{1\le j\le n_i}\Gamma_{i,j}$, $L(v_i) = \bigcup_{1\le j\le n_i}\{q_{i,j}\}$ and $L(V_{i+1}) = \bigl\{L(v_{i+1})\bigr\} = \bigl\{\iota\bigl(\bigcup_{1\le j\le n_i} Q'_{i,j}\bigr)\bigr\}$ hold, the labeling of the node v_{i+1} and the edge $\langle v_i, V_{i+1}\rangle$ is consistent. It follows that the assumptions used in the inductive construction hold at level $i+1$ of G, and we may repeat the construction. This completes the inductive definition of L and the integers c_i; by the inductive argument, it follows that the graph $G = \langle V, E, L\rangle$ is a run of $\mathcal{A}_\iota$ on w.

---

[3] As in Sect. 4.3.3, we define $I_{\mathcal{A},f}(w') \overset{def}{=} \bigl\{0 \le i < \omega \,\big|\, \mathcal{A}^{q_f} \text{ fin-accepts } (w')^i \text{ by avoiding an initial } f\text{-transition}\bigr\}$ for all $f \in \mathcal{F}$ and $w' \in (2^{AP})^\omega$.

The reasoning used in the proof of Proposition 4.3.5 applies directly to show that $c_i = f$ holds for infinitely many indices $0 \leq i < \omega$ for all acceptance conditions $f \in \mathcal{F}$. Similarly, if $I_{\mathcal{A},f}(w) < \omega$ holds for an acceptance condition $f \in \mathcal{F}$, then there exists an index $0 \leq i < \omega$ such that no state in $L(v_j)$ is an f-state for all $i \leq j < \omega$ (Lemma 4.3.6), and thus (because the transition starting from the node v_j is formed from transitions of $\mathcal{A}$ whose source states are in $L(v_j)$) G is fin-acceptance synchronized with respect to the condition f. Otherwise $I_{\mathcal{A},f}(w) = \omega$ holds, in which case the inductive construction of L guarantees that $L\big(\langle v_i, V_{i+1}\rangle\big)$ is not an f-transition for infinitely many $0 \leq i < \omega$, and G is fin-acceptance synchronized with respect to f also in this case. Because the same result holds for all acceptance conditions, it follows that G is fin-acceptance synchronized, and therefore fin-accepting (Proposition 4.3.1). We conclude that $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_\iota)$ holds. $\square$

Proposition 4.5.2 now follows from Lemma 4.5.4 and Lemma 4.5.5.

The Subautomaton $\mathcal{A}^{q_I}$

Proposition 4.5.2 shows that the state set of the nondeterministic automaton obtained using the universal subset construction from an alternating automaton $\mathcal{A}$ built from an LTL formula $\varphi \in LTL^{\text{PNF}}(AP)$ using the translation rules of Sect. 3.1 can be reduced by, in effect, merging states that have identical images under a function ι satisfying certain constraints; the function $\mathcal{SI}$ defined in Sect. 4.5.1 provides a concrete example of such a function.

In practice, however, we are usually not interested in the full automaton $\mathcal{A}$, but instead the subautomaton $\mathcal{A}^{q_I}$ rooted at its initial state, and a nondeterministic automaton obtained from $\mathcal{A}^{q_I}$. As observed in Sect. 3.2.1, the state set Q^{q_I} of $\mathcal{A}^{q_I}$ consists of those states of $\mathcal{A}$ that correspond to (i) the formula φ itself, (ii) a binary pure temporal subformula of φ, or (iii) a subformula $\psi \in \text{NSub}(\varphi)$ such that $\mathsf{X}\psi \in \text{NSub}(\varphi)$ holds. Consequently, all states reachable from the initial state $\{q_I\}$ of the nondeterministic automaton obtained from $\mathcal{A}$ using the universal subset construction of Theorem 4.3.2 correspond to conjunctions of these formulas. However, this correspondence is not preserved in the nondeterminization construction of Proposition 4.5.2 when using the function $\mathcal{SI}$ to identify equivalent states of the nondeterministic automaton: in the general case, there may exist subsets $Q' \subseteq Q^{q_I}$ for which $\mathcal{SI}(Q') \subseteq Q^{q_I}$ does not hold. For example, the subautomaton rooted at the initial state of the nondeterministic automaton built for the formula $(p_1 \mathsf{R_s}\, p_2)$ by applying Theorem 4.3.2 consists of a single state that corresponds to the set of binary pure temporal formulas $\big\{(p_1 \mathsf{R_s}\, p_2)\big\}$, but the initial state of the automaton built using the construction of Proposition 4.5.2 corresponds to the set of formulas $\mathcal{SI}\big(\big\{(p_1 \mathsf{R_s}\, p_2)\big\}\big) = \big\{(p_1 \mathsf{R_s}\, p_2), p_2, \top\big\}$, which contains also Boolean formulas. In more complex cases, it is not even apparent (without further analysis of the function $\mathcal{SI}$) from the nondeterminization construction of Proposition 4.5.2 why the construction could not in fact yield a nondeterministic automaton having more states reachable from its initial state than the automaton built using the standard universal subset construction. Thus it is not obvious whether the upper bounds given in Corollary 4.3.9 still hold for nondeterministic automata built using the construction of Proposition 4.5.2.

The above technical difficulty can be avoided via a simple change to the function $\mathcal{SI}$ to make it closed with respect to the set $2^{Q^{q_I}}$ by simply projecting the image $\mathcal{SI}(Q')$ of any subset $Q' \subseteq Q^{q_I}$ back to a subset of Q^{q_I}. That is, we may refine the function $\mathcal{SI}$ used in the nondeterminization construction into a function $\mathcal{SI}' : 2^Q \to 2^Q$ by defining $\mathcal{SI}'(Q') \stackrel{def}{=} \mathcal{SI}(Q') \cap Q^{q_I}$ for all $Q' \subseteq Q^{q_I}$ (and $\mathcal{SI}'(Q') \stackrel{def}{=} \mathcal{SI}(Q')$ for all $2^Q \setminus 2^{Q^{q_I}}$); it is straightforward to check that the function $\mathcal{SI}'$ still has the properties required of a function usable in the construction for every subset $Q' \subseteq Q$. The states in the subautomaton rooted at the initial state of the nondeterministic automaton built by using the function $\mathcal{SI}'$ for nondeterminization can thus be seen to be elements of the set $\{\mathcal{SI}'(Q') \mid Q' \subseteq Q^{q_I}\} \subseteq 2^{Q^{q_I}}$. The size of this set does not exceed the upper bound given in Corollary 4.3.9: as in the proof of Corollary 4.2.2 (p. 79), we can consider the initial state of $\mathcal{A}^{q_I}$ to be an element separate from Q^{q_I} if the formula φ is not a binary pure temporal formula. As a further possible advantage of using the function $\mathcal{SI}'$ instead of the function $\mathcal{SI}$ in the construction, the fact that the values of $\mathcal{SI}'$ are subsets of the corresponding values of $\mathcal{SI}$ may speed up the construction if the transitions of the nondeterministic automaton are to be built explicitly: the number of possible combinations of transitions starting from a state in the nondeterministic automaton is in the worst case exponential in the number of state components (i.e., the size of the state when seen as a subset of Q).

4.6 THE SUBCLASS LTL^{CND}

We end this chapter by investigating a syntactic subclass of LTL for which translation into self-loop alternating automata can be combined with a simple completion lemma to obtain a direct translation procedure from this subclass of LTL into nondeterministic automata. Essentially the same subclass of LTL has previously been considered by Schneider [1999] in the context of symbolic translation algorithms from LTL into automata. A closely related approach was also taken by Maidl [2000a], who investigated translation of another syntactic subclass of LTL into one-weak nondeterministic automata. Although these two subclasses of LTL have different syntactic definitions and use different translation rules, we shall show that the LTL subclass used by Maidl actually coincides with an explicit version of the subclass of Schneider. This version can be extracted from the translation procedure presented in Sect. 3.1 by studying the closure properties of the rules. Because any formula from this subclass can be translated into a self-loop nondeterministic automaton with a set of states whose size is linear in the number of pure temporal subformulas in the formula, each satisfiable formula in the subclass has a model that can be represented in polynomial space in the length of the formula. It follows that the decision problem of testing the satisfiability of formulas in the subclass is $\mathbf{NP}$-complete.

4.6.1 Completion to Nondeterministic Automata

By definition, a nondeterministic automaton is an alternating automaton, all transitions of which have exactly one target state. Obviously, alternating automata having no transitions with two or more target states "almost" satisfy this condition with the possible exception of transitions whose target state set is empty. It is straightforward to make these automata nondeterministic by redirecting all such transitions to an additional "sink" state, the subautomaton rooted at which always accepts its input.[4]

Lemma 4.6.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton such that $|Q'| \leq 1$ holds for all transitions $\langle q, \Gamma, F, Q' \rangle \in \Delta$. Let $\hat{q}$ be a new state not included in Q. The automaton $\mathcal{A}' = \big\langle \Sigma, Q \cup \{\hat{q}\}, \Delta', q_I, \mathcal{F} \big\rangle$, where $\Delta' \stackrel{def}{=} \big\{ \langle q, \Gamma, F, Q' \rangle \in \Delta \,\big|\, Q' \neq \emptyset \big\} \cup \big\{ \langle q, \Gamma, F, \{\hat{q}\} \rangle \,\big|\, \langle q, \Gamma, F, \emptyset \rangle \in \Delta \big\} \cup \big\{ \langle \hat{q}, \Sigma, \emptyset, \{\hat{q}\} \rangle \big\}$, is a nondeterministic automaton with $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}) = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}')$, and $\mathcal{A}'$ is a self-loop automaton iff $\mathcal{A}$ is.*

Proof: Because $|Q'| \leq 1$ holds for all $\langle q, \Gamma, F, Q' \rangle \in \Delta$, it is easy to see from the definition of $\mathcal{A}'$ that every transition of $\mathcal{A}'$ has exactly one target state, and thus $\mathcal{A}'$ is nondeterministic. Because the definition obviously preserves all cycles of $\mathcal{A}$ and adds only a self-loop to the automaton, $\mathcal{A}'$ is a self-loop nondeterministic automaton if $\mathcal{A}$ is a self-loop alternating automaton and vice versa. We check that $\mathcal{A}$ and $\mathcal{A}'$ fin-accept the same language.

$(\mathcal{L}_{\mathrm{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}'))$ Let $w \in \Sigma^\omega$ be a word fin-accepted by the automaton $\mathcal{A}$, and let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}$ on w. Define the graph $G' = \langle V', E', L' \rangle$, where $V' \stackrel{def}{=} V$, $E' \stackrel{def}{=} \big\{ \langle v, V' \rangle \in E \,\big|\, V' \neq \emptyset \big\}$, and $L'(x) \stackrel{def}{=} L(x)$ for all $x \in V' \cup E'$. Because G is a run and $V' \subseteq V$ and $E' \subseteq E$, V'_0 is a singleton, and V' is obviously partitioned into disjoint finite levels such that E' contains edges only between successive levels of G'. For the same reason, each node of V' has at most one outgoing edge, and each node $v' \in V'_i$ $(1 \leq i < \omega)$ is a successor of another node in G' (it is a successor of another node in G, and E' includes all edges in E with a nonempty set of target nodes). Furthermore, the labeling of v_0 and each edge in E' is consistent. Finally, because all infinite branches in G' are obviously infinite branches of the fin-accepting run G, the branches satisfy the fin-acceptance condition, and it follows that G' is a fin-accepting semi-run of $\mathcal{A}'$ on w.

Let $v \in V'_i$ be a node with no outgoing edges for some $0 \leq i < \omega$. In G, this node has the unique outgoing edge $e \in E$ with an empty set of target nodes such that $L(e) = \big\langle L(v), \Gamma, F, \emptyset \big\rangle \in \Delta$ holds for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$, and $w(i) \in \Gamma$. By the definition of $\mathcal{A}'$, $(\mathcal{A}')^{L'(v)} = (\mathcal{A}')^{L(v)}$ now has an initial transition $\big\langle L'(v), \Gamma, F, \{\hat{q}\} \big\rangle \in \Delta'$. Furthermore, because $\mathcal{L}_{\mathrm{fin}}\big((\mathcal{A}')^{\hat{q}}\big) = \Sigma^\omega$ obviously holds, it follows that $(\mathcal{A}')^{\hat{q}}$ fin-accepts w^{i+1}, and therefore $(\mathcal{A}')^{L'(v)}$ fin-accepts w^i by Proposition 2.3.15. Since this same result holds for all nodes of V' with no outgoing edges, we can apply Proposition 2.3.14 to extend the semi-run G' into a fin-accepting run of $\mathcal{A}'$ on w, and thus $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}')$ holds.

$(\mathcal{L}_{\mathrm{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}))$ Let $G' = \langle V', E', L' \rangle$ be a fin-accepting run of $\mathcal{A}'$ on some $w \in \Sigma^\omega$. In this case we can define a fin-accepting semi-run $G = \langle V, E, L \rangle$ of $\mathcal{A}$ on w as follows:

[4] Alternatively, we could simply relax the definition of nondeterministic automata to permit transitions with no target states. We nevertheless use the more traditional definition of nondeterministic automata, which also saves us from the possible need to consider such transitions a special case when working with nondeterministic automata.

- $V \stackrel{def}{=} \{v \in V' \,|\, L'(v) \neq \hat{q}\}$;
- $E \stackrel{def}{=} \{e \in E' \,|\, L'(e) = \langle q, \Gamma, F, Q' \rangle \in \Delta', \hat{q} \notin \{q\} \cup Q'\}$; and
- $L(x) \stackrel{def}{=} L'(x)$ for all $x \in V \cup E$.

As above, because $V \subseteq V'$ and $E \subseteq E'$, V can be partitioned into finite disjoint levels such that V_0 is a singleton (because $\hat{q} \notin L'(V_0) = \{q_I\}$ holds), the edges of G lie between successive levels of V, and since $E \subseteq E'$, each node of v has at most one outgoing edge, and the labeling L is consistent for all $x \in V \cup E$.

If there exists a node $v' \in V \setminus V_0$ that is not a successor of another node in V, then, because G' is a run, there exists a node $v \in V'$ and an edge $e' = \langle v, V'' \rangle \in E'$ that includes v' in its target nodes. Because $e' \notin E$, however, it follows that the transition $L'(e') \in \Delta'$ has $\hat{q}$ either as its source state or as one of its target states. By the definition of Δ', $\hat{q}$ is actually the only target state of $L'(e')$, and thus all nodes of V'' are labeled with $\hat{q}$ in G'. In particular, $L'(v') = \hat{q}$ holds. This contradicts the fact that $v' \in V$, however. Therefore v' is necessarily a successor of another node in V.

Because the infinite branches of G form a subset of the infinite branches of G' (all of which satisfy the fin-acceptance condition), it follows that G is a fin-accepting semi-run of $\mathcal{A}$ on w.

If $v \in V_i$ is a node in G with no outgoing edges for some $0 \leq i < \omega$, then $L(v) = L'(v) \neq \hat{q}$, and there exists an edge $e \in E'$ labeled with a transition $\langle L'(v), \Gamma, F, \{\hat{q}\} \rangle \in \Delta'$ for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w(i) \in \Gamma$ holds. Because $L'(v) = L(v)$ holds, it follows from the definition of $\mathcal{A}'$ that $\langle L(v), \Gamma, F, \emptyset \rangle \in \Delta$ is an initial transition of $\mathcal{A}^{L(v)}$. Because the target state set of this transition is empty, Proposition 2.3.15 applies trivially, and thus $\mathcal{A}^{L(v)}$ fin-accepts w^i. Because v is arbitrary, it follows by Proposition 2.3.14 that G can be extended into a fin-accepting run of $\mathcal{A}$ on w, and therefore $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds. □

4.6.2 Closure Properties of Translation Rules

Lemma 4.6.1 makes it possible to complete any alternating automaton having no transitions with two or more target states into a nondeterministic automaton that fin-recognizes its language. Our goal is to identify a class of LTL formulas which admit translation into alternating automata that can be made nondeterministic simply by applying this lemma. A natural approach is to consider the *closure* properties of the translation rules (cf. [Schneider 1999]), i.e., the conditions under which an automaton built using a translation rule can be made nondeterministic using Lemma 4.6.1 provided that the automata to which the rule was applied had this property.

Clearly, an alternating automaton built from an atomic formula has at most one transition, and this transition has no target states. Hence, this automaton can be completed into a nondeterministic automaton by applying Lemma 4.6.1. The other translation rules have the following simple closure properties.

Lemma 4.6.2 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two self-loop alternating automata having no transitions with two or more distinct target states, and let $\mathcal{A}_3$ be a single-state alternating automaton, all transitions of which have an empty set of target states. All automata built from*

(a) $\mathcal{A}_1$ with the translation rule for the X *operator,*

(b) $\mathcal{A}_1$ and $\mathcal{A}_2$ (or $\mathcal{A}_2$ and $\mathcal{A}_1$) with the translation rule for the $\vee$ operator,

(c) $\mathcal{A}_1$ and $\mathcal{A}_3$ (or $\mathcal{A}_3$ and $\mathcal{A}_1$) with the translation rule for the $\wedge$ operator,

(d) $\mathcal{A}_3$ and $\mathcal{A}_1$ with one of the U *translation rules (using the initial transitions of $\mathcal{A}_3$ to define the initial self-loops of the compound automaton)*

(e) $\mathcal{A}_1$ and $\mathcal{A}_3$ with one of the R *translation rules (using the initial transitions of $\mathcal{A}_3$ to define the initial self-loops of the compound automaton)*

are self-loop alternating automata with no transitions having two or more distinct target states.

Proof: It is clear by the discussion in Sect. 3.1.1 that an automaton built by applying a translation rule to self-loop alternating automata is itself a self-loop alternating automaton. We verify the claim that this automaton has no transitions with two or more distinct target states.

(a) Applying the X translation rule to the automaton $\mathcal{A}_1$ adds one transition to this automaton. Because the only target state of this transition is the initial state of $\mathcal{A}_1$, the result follows from the assumption that all transitions of $\mathcal{A}_1$ have at most one target state.

(b) The translation rule for the $\vee$ operator creates transitions, each of which shares its target states with some initial transition of $\mathcal{A}_1$ or $\mathcal{A}_2$. The result now holds again by the assumption that $\mathcal{A}_1$ and $\mathcal{A}_2$ have no transitions with two or more distinct target states.

(c) By the assumption, the target state set of each transition of $\mathcal{A}_3$ is empty. Thus, because all transitions of $\mathcal{A}_1$ have at most one target state, the union of the target state sets of any pair of transitions of $\mathcal{A}_1$ and $\mathcal{A}_3$ is a set containing at most one state. The result now follows from the properties of $\mathcal{A}_1$ and $\mathcal{A}_3$ and the fact that the target state sets of all transitions created by the $\wedge$ translation rule are formed in this way.

(d) Each transition created by one of the translation rules for the U connectives either shares its target states with some initial transition of $\mathcal{A}_1$, or its target states are formed by adding the initial state of the constructed automaton to the (empty) target state set of some initial transition of $\mathcal{A}_3$. Because no transition of $\mathcal{A}_1$ has two or more distinct target states, there are no such transitions in the compound automaton, either.

(e) The translation rules for the R connectives form the target states of each new transition either by augmenting the (empty) target state set of one of the initial transitions of $\mathcal{A}_3$ with the initial state of the constructed automaton, or by collecting the target states of a pair of $\mathcal{A}_1$'s and $\mathcal{A}_3$'s initial transitions. It is easy to see from the properties of $\mathcal{A}_1$ and $\mathcal{A}_3$ that no transition of the compound automaton will have two or more target states. $\square$

4.6.3 Definition of the Subclass

Lemma 4.6.2 helps to isolate a simple syntactic fragment of LTL that can be translated into self-loop alternating automata having no transitions with two or more target states. Formally, for a countable set AP of atomic propositions, we define the set $LTL^{\mathrm{CND}}(AP)$ ("subset of LTL for which the translation rules are closed under translation into nondeterministic automata") as the smallest set of formulas that is closed under the finite application of the syntactic rule

$$\text{If } \theta \in PL(AP) \text{ and } \varphi_1, \varphi_2 \in LTL^{\mathrm{CND}}(AP)\text{, then}$$
$$\theta, \mathsf{X}\varphi_1, (\varphi_1 \vee \varphi_2), (\varphi_1 \wedge \theta), (\theta \wedge \varphi_1), (\theta \,\mathsf{U}\, \varphi_1), (\varphi_1 \,\mathsf{R}\, \theta) \in LTL^{\mathrm{CND}}(AP)$$

(where U and R can be either weak or strong binary temporal connectives).

It is easy to see that $[\varphi]^{\mathrm{PNF}} \in LTL^{\mathrm{CND}}(AP)$ holds for all formulas $\varphi \in LTL^{\mathrm{CND}}(AP)$, because the syntactic definition allows negation to occur only in propositional subformulas of φ (and the positive normal form of such a subformula is also a propositional formula). Obviously, $|\mathrm{Temp}(\varphi)| = \big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big|$ then holds also.

Proposition 4.6.3 *Let $\varphi \in LTL^{\mathrm{CND}}(AP)$. The self-loop alternating automaton built from $[\varphi]^{\mathrm{PNF}}$ using the translation rules presented in Sect. 3.1 has no transitions with two or more distinct target states.*

Proof: Because $\varphi \equiv [\varphi]^{\mathrm{PNF}}$ and $[\varphi]^{\mathrm{PNF}} \in LTL^{\mathrm{CND}}(AP)$ hold, we may, without loss of generality, assume that φ itself is in positive normal form. It is clear from the basic translation rules that the result holds if φ is an atomic formula. Assume that the result holds for all formulas in $LTL^{\mathrm{CND}}(AP)$ of length less than or equal to some $1 \leq k < \omega$, and assume that φ is a non-atomic formula of length $k + 1$. Due to the minimality of $LTL^{\mathrm{CND}}(AP)$, φ is a compound formula of one of the forms presented in the definition of the subclass, and therefore φ has one or two top-level subformulas in $LTL^{\mathrm{CND}}(AP)$ of length at most k. By the induction hypothesis, these subformulas can be translated into self-loop alternating automata with no transitions having two or more target states. The result now follows by induction, where the induction step can be proved for each different kind of compound formula using Lemma 4.6.2.

For example, if φ is of the form $(\theta \,\mathsf{U}\, \varphi_1)$ for some $\theta \in PL(AP)$ and $\varphi_1 \in LTL^{\mathrm{CND}}(AP)$, then there exist self-loop alternating automata having the desired property for the formulas θ and φ_1 by the induction hypothesis. More precisely, because θ is a propositional formula, θ has a corresponding self-loop alternating automaton, all transitions of which have an empty set of target states. By Lemma 4.6.2 (d), it now follows that the compound automaton built from these automata using one of the translation rules for the U connectives has no transitions with two or more target states. □

By combining Proposition 4.6.3 and Theorem 3.3.2 with Lemma 4.6.1, we obtain an effective procedure for translating formulas from $LTL^{\mathrm{CND}}(AP)$ directly into nondeterministic automata. It follows that the number of states in an automaton that recognizes the language of a formula in the subclass $LTL^{\mathrm{CND}}(AP)$ depends linearly on the number of syntactically distinct pure temporal subformulas in the formula.

Corollary 4.6.4 *Let $\varphi \in LTL^{\mathrm{CND}}(AP)$ be an LTL formula. The language $\mathcal{L}(\varphi)$ can be fin-recognized by a self-loop nondeterministic automaton with at most $2 + |\mathrm{Temp}(\varphi)|$ states.*

Proof: Let $\mathcal{A}$ be a self-loop alternating automaton built from $[\varphi]^{\mathrm{PNF}}$ using the basic translation rules, let q_I be the initial state of the automaton, and let Q^{q_I} be the state set of the subautomaton $\mathcal{A}^{q_I}$. By Proposition 4.6.3, $\mathcal{A}$ has no transitions with two or more target states, and $\mathcal{A}^{q_I}$ has this same property. If $\mathcal{A}$ is not nondeterministic, then $\mathcal{A}$ can be completed to an equivalent nondeterministic automaton by applying Lemma 4.6.1. This modification adds a new state to the automaton. The result then follows because $|Q^{q_I}| \leq 1 + \big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big| = 1 + |\mathrm{Temp}(\varphi)|$ (Corollary 3.2.2) and because $\mathcal{A}$ and $\mathcal{A}^{q_I}$ fin-accept the same language (Proposition 2.3.12). □

4.6.4 Relationships between Syntactic Subclasses of LTL

As a closely related approach, Maidl [2000a] defined a subclass of LTL called LTL$^{\mathrm{det}}$ that consists of formulas whose negation can be translated into self-loop nondeterministic automata. (The class LTL$^{\mathrm{det}}$ itself corresponds to the set of LTL formulas, for each of which there exists a formula in the branching time logic ∀CTL such that the model checking problems for these formulas in any structure have the same solution [Maidl 2000a].) We denote the class of negations of formulas in LTL$^{\mathrm{det}}$ (over a given set AP of atomic propositions) by $LTL^{\overline{\mathrm{det}}}(AP)$; this subclass of LTL can be defined directly as the smallest subset of $LTL(AP)$ that is closed under the finite application of the syntactic rule

$$\begin{gathered}\text{If } \theta \in PL(AP) \text{ and } \varphi_1, \varphi_2 \in LTL^{\overline{\mathrm{det}}}(AP)\text{, then}\\ \theta, \mathsf{X}\varphi_1, (\varphi_1 \vee \varphi_2), \big((\theta \vee \varphi_1) \wedge (\neg\theta \vee \varphi_2)\big)\text{, and}\\ \big((\theta \vee \varphi_1)\, \mathsf{R}\, (\neg\theta \vee \varphi_2)\big) \in LTL^{\overline{\mathrm{det}}}(AP)\end{gathered}$$

(where R can be either of the Release connectives).

The relationship between the expressive power of the syntactic subclasses $LTL^{\mathrm{CND}}(AP)$ and $LTL^{\overline{\mathrm{det}}}(AP)$ is not immediately obvious from the syntactic definitions of the subclasses. For example, the subclass $LTL^{\overline{\mathrm{det}}}(AP)$ includes binary pure temporal formulas, both top-level subformulas of which are temporal formulas. As we shall show below, the subclasses in fact describe the same LTL properties equally succinctly in the number of syntactically distinct pure temporal subformulas. We first define sets of recursive rules for rewriting formulas in $LTL^{\mathrm{CND}}(AP)$ and $LTL^{\overline{\mathrm{det}}}(AP)$. We then show that the rules define mappings between $LTL^{\mathrm{CND}}(AP)$ and $LTL^{\overline{\mathrm{det}}}(AP)$ that preserve the logical equivalence of formulas (Proposition 4.6.6). Finally, we show that the mappings do not increase the number of syntactically distinct pure temporal subformulas (Proposition 4.6.9).

Translations between $LTL^{\mathrm{CND}}(AP)$ and $LTL^{\overline{\mathrm{det}}}(AP)$

Let $\varphi \in LTL^{\mathrm{CND}}(AP)$. We associate with φ the formula $[\varphi]^{\overline{\mathrm{det}}}$ obtained from φ by applying the following recursive rewrite rules (where $\varphi_1, \varphi_2 \in$

$LTL^{\mathrm{CND}}(AP)$ and $\theta \in PL(AP)$):

$$
\begin{aligned}
[\varphi]^{\overline{\det}} &\stackrel{\text{def}}{=} \varphi \qquad \text{if } \varphi \in PL(AP)\text{, and for formulas not in } PL(AP),\\
[\mathsf{X}\varphi_1]^{\overline{\det}} &\stackrel{\text{def}}{=} \mathsf{X}[\varphi_1]^{\overline{\det}}\\
\big[(\varphi_1 \vee \varphi_2)\big]^{\overline{\det}} &\stackrel{\text{def}}{=} \big([\varphi_1]^{\overline{\det}} \vee [\varphi_2]^{\overline{\det}}\big)\\
\big[(\varphi_1 \wedge \theta)\big]^{\overline{\det}} &\stackrel{\text{def}}{=} \big((\theta \vee \bot) \wedge (\neg\theta \vee [\varphi_1]^{\overline{\det}})\big)\\
\big[(\theta \wedge \varphi_1)\big]^{\overline{\det}} &\stackrel{\text{def}}{=} \big((\theta \vee \bot) \wedge (\neg\theta \vee [\varphi_1]^{\overline{\det}})\big)\\
\big[(\theta \,\mathsf{U}\, \varphi_1)\big]^{\overline{\det}} &\stackrel{\text{def}}{=} \big((\neg\theta \vee [\varphi_1]^{\overline{\det}}) \,\mathsf{R}\, (\neg(\neg\theta) \vee [\varphi_1]^{\overline{\det}})\big)\\
\big[(\varphi_1 \,\mathsf{R}\, \theta)\big]^{\overline{\det}} &\stackrel{\text{def}}{=} \big((\neg\theta \vee [\varphi_1]^{\overline{\det}}) \,\mathsf{R}\, (\neg(\neg\theta) \vee \bot)\big)
\end{aligned}
$$

Conversely, we also associate with each formula $\varphi \in LTL^{\overline{\det}}(AP)$ the formula $[\varphi]^{\mathrm{CND}}$ defined recursively as follows:

$$
\begin{aligned}
[\varphi]^{\mathrm{CND}} &\stackrel{\text{def}}{=} \varphi \qquad \text{if } \varphi \in PL(AP)\text{; and otherwise}\\
[\mathsf{X}\varphi_1]^{\mathrm{CND}} &\stackrel{\text{def}}{=} \mathsf{X}[\varphi_1]^{\mathrm{CND}}\\
\big[(\varphi_1 \vee \varphi_2)\big]^{\mathrm{CND}} &\stackrel{\text{def}}{=} \big([\varphi_1]^{\mathrm{CND}} \vee [\varphi_2]^{\mathrm{CND}}\big)\\
\big[((\theta \vee \varphi_1) \wedge (\neg\theta \vee \varphi_2))\big]^{\mathrm{CND}} &\stackrel{\text{def}}{=} \big((\neg\theta \wedge [\varphi_1]^{\mathrm{CND}}) \vee (\theta \wedge [\varphi_2]^{\mathrm{CND}})\big)\\
\big[((\theta \vee \varphi_1) \,\mathsf{R}\, (\neg\theta \vee \varphi_2))\big]^{\mathrm{CND}} &\stackrel{\text{def}}{=} \big(\neg\theta \,\mathsf{U}\, ((\neg\theta \wedge [\varphi_1]^{\mathrm{CND}}) \vee (\theta \wedge [\varphi_2]^{\mathrm{CND}}))\big)
\end{aligned}
$$

(In rules involving binary temporal connectives, the binary temporal connectives always have the same strength on both sides of the definition.)

Expressive Equivalence of $LTL^{\mathrm{CND}}(AP)$ and $LTL^{\overline{\det}}(AP)$

To show that the above rewrite rules define mappings between the subclasses $LTL^{\mathrm{CND}}(AP)$ and $LTL^{\overline{\det}}(AP)$ such that the mappings preserve the logical equivalence of formulas, we first list the following simple LTL identities for reference.

Lemma 4.6.5 *The following identities hold for all LTL formulas $\varphi_1, \varphi_2 \in LTL(AP)$ (where the temporal connectives in an identity are of the same strength on both sides of the identity):*

(a) $(\varphi_1 \,\mathsf{U}\, \varphi_2) \equiv \big((\varphi_1 \vee \varphi_2) \,\mathsf{U}\, \varphi_2\big)$;

(b) $(\varphi_1 \,\mathsf{U}\, \varphi_2) \equiv \big((\neg\varphi_1 \vee \varphi_2) \,\mathsf{R}\, (\varphi_1 \vee \varphi_2)\big)$;

(c) $(\varphi_1 \,\mathsf{R}\, \varphi_2) \equiv \big((\neg\varphi_2 \vee \varphi_1) \,\mathsf{R}\, \varphi_2\big)$.

<u>Proof:</u> Let $w \in (2^{AP})^\omega$. We check that the identities hold by the semantics of LTL.

<u>(a)</u> $w \models (\varphi_1 \,\mathsf{U}\, \varphi_2)$

iff there exists an index $0 \leq i < \omega$ such that $w^i \models \varphi_2$ holds, and $w^j \models \varphi_1$ holds for all $0 \leq j < i$ (or if $\mathsf{U} = \mathsf{U}_\mathsf{w}$, and $w^i \models \varphi_1$ holds for all $0 \leq i < \omega$) *(semantics of U)*

iff there exists a least index $0 \leq i < \omega$ such that $w^i \models \varphi_2$ holds, and $w^j \models \varphi_1$ holds for all $0 \leq j < i$ (or if $\mathsf{U} = \mathsf{U}_\mathsf{w}$, and $w^i \models \varphi_2$ does not hold for any, but $w^i \models \varphi_1$ holds for all $0 \leq i < \omega$)

iff there exists a least index $0 \le i < \omega$ such that $w^i \models \varphi_2$ holds, and $w^j \models (\varphi_1 \vee \varphi_2)$ holds for all $0 \le j < i$ (or if $\mathsf{U} = \mathsf{U_w}$, and $w^i \models \varphi_2$ does not hold for any, but $w^i \models (\varphi_1 \vee \varphi_2)$ holds for all $0 \le i < \omega$)

iff there exists an index $0 \le i < \omega$ such that $w^i \models \varphi_2$ holds, and $w^j \models (\varphi_1 \vee \varphi_2)$ holds for all $0 \le j < i$ (or if $\mathsf{U} = \mathsf{U_w}$, and $w^i \models (\varphi_1 \vee \varphi_2)$ holds for all $0 \le i < \omega$)

iff $w \models \big((\varphi_1 \vee \varphi_2) \,\mathsf{U}\, \varphi_2\big)$. *(semantics of* U*)*

(b) $w \models (\varphi_1 \,\mathsf{U}\, \varphi_2)$

iff $w \models \big((\varphi_1 \vee \varphi_2) \,\mathsf{U}\, \varphi_2\big)$ *(by (a))*

iff $w \models \big((\varphi_1 \vee \varphi_2) \,\mathsf{U}\, (\bot \vee \varphi_2)\big)$ $(\varphi_2 \equiv (\bot \vee \varphi_2))$

iff $w \models \big((\varphi_1 \vee \varphi_2) \,\mathsf{U}\, ((\varphi_1 \wedge \neg\varphi_1) \vee \varphi_2)\big)$ $(\bot \equiv (\varphi_1 \wedge \neg\varphi_1))$

iff $w \models \big((\varphi_1 \vee \varphi_2) \,\mathsf{U}\, ((\varphi_1 \vee \varphi_2) \wedge (\neg\varphi_1 \vee \varphi_2))\big)$
$\big(((\varphi_1 \wedge \neg\varphi_1) \vee \varphi_2) \equiv ((\varphi_1 \vee \varphi_2) \wedge (\neg\varphi_1 \vee \varphi_2))\big)$

iff $w \models \big((\neg\varphi_1 \vee \varphi_2) \,\mathsf{R}\, (\varphi_1 \vee \varphi_2)\big)$. *(definition of* R *in terms of* U*)*

(c) $w \models (\varphi_1 \,\mathsf{R}\, \varphi_2)$

iff $w \models \big(\varphi_2 \,\mathsf{U}\, (\varphi_1 \wedge \varphi_2)\big)$ *(definition of* R *in terms of* U*)*

iff $w \models \big(\varphi_2 \,\mathsf{U}\, ((\neg\varphi_2 \vee \varphi_1) \wedge \varphi_2)\big)$ $\big((\varphi_1 \wedge \varphi_2) \equiv ((\neg\varphi_2 \vee \varphi_1) \wedge \varphi_2)\big)$

iff $w \models \big((\neg\varphi_2 \vee \varphi_1) \,\mathsf{R}\, \varphi_2\big)$. *(definition of* R *in terms of* U*)*

$\square$

It is now straightforward to show that $[\,\cdot\,]^{\overline{\text{det}}}$ is a mapping from the subclass $LTL^{\text{CND}}(AP)$ to $LTL^{\overline{\text{det}}}(AP)$ that preserves the logical equivalence of formulas (and conversely for $[\,\cdot\,]^{\text{CND}}$ in the opposite direction).

Proposition 4.6.6 *Let $\mathcal{C}$ be one of the syntactic subclasses $LTL^{\text{CND}}(AP)$ or $LTL^{\overline{\text{det}}}(AP)$, let $\mathcal{C}'$ be the opposite subclass, let $\varphi \in \mathcal{C}$, and let $[\varphi]$ be the formula obtained from φ via the translation defined above for formulas in the subclass $\mathcal{C}$. The formulas φ and $[\varphi]$ are logically equivalent, and $[\varphi] \in \mathcal{C}'$ holds.*

Proof: The result obviously holds if φ is a propositional formula, because $PL(AP) \subseteq LTL^{\text{CND}}(AP) \cap LTL^{\overline{\text{det}}}(AP)$. In particular, the result holds if $|\varphi| = 1$.

Assume that $\varphi \equiv [\varphi] \in \mathcal{C}'$ holds for all formulas $\varphi \in \mathcal{C}$ of length at most $1 \le k < \omega$, and let $\varphi \in \mathcal{C}$ be a temporal formula of length $k+1$. If $\varphi = \mathsf{X}\varphi_1$ or $\varphi = (\varphi_1 \vee \varphi_2)$ holds for some $\varphi_1, \varphi_2 \in \mathcal{C}$ ($|\varphi_1| \le k$, $|\varphi_2| \le k$), then either $[\varphi] = \mathsf{X}[\varphi_1]$ or $[\varphi] = \big([\varphi_1] \vee [\varphi_2]\big)$, and the result follows easily by the semantics of LTL and the syntactic closure properties of the subclass $\mathcal{C}'$ since $[\varphi_1] \equiv \varphi_1 \in \mathcal{C}'$ and $[\varphi_2] \equiv \varphi_2 \in \mathcal{C}'$ hold by the induction hypothesis. We check the other possible cases in each direction. Below, φ_1 and φ_2 are formulas in the subclass $\mathcal{C}$, and θ is a propositional formula.

($\mathcal{C} = LTL^{\text{CND}}(AP), \mathcal{C}' = LTL^{\overline{\text{det}}}(AP)$) If $\varphi = (\varphi_1 \wedge \theta)$ or $\varphi = (\theta \wedge \varphi_1)$ holds, then it is easy to check from the semantics of LTL that $\varphi \equiv (\theta \wedge \varphi_1) \equiv \big(\theta \wedge (\neg\theta \vee \varphi_1)\big) \equiv \big((\theta \vee \bot) \wedge (\neg\theta \vee \varphi_1)\big)$ holds.

If $\varphi = (\theta \mathsf{U} \varphi_1)$, then $\varphi = (\theta \mathsf{U} \varphi_1) \equiv \big((\neg\theta \vee \varphi_1) \mathsf{R} (\theta \vee \varphi_1)\big) \equiv \big((\neg\theta \vee \varphi_1) \mathsf{R} (\neg(\neg\theta) \vee \varphi_1)\big)$ holds by Lemma 4.6.5 (b) and the semantics of LTL.

Finally, if $\varphi = (\varphi_1 \mathsf{R} \theta)$, then $\varphi = (\varphi_1 \mathsf{R} \theta) \equiv \big((\neg\theta \vee \varphi_1) \mathsf{R} \theta\big) \equiv \big((\neg\theta \vee \varphi_1) \mathsf{R} (\theta \vee \bot)\big) \equiv \big((\neg\theta \vee \varphi_1) \mathsf{R} (\neg(\neg\theta) \vee \bot)\big)$ holds by Lemma 4.6.5 (c) (used in the first logical equivalence) and the semantics of LTL.

In each case, we obtain the formula $[\varphi]$ from the last formula in the chain of logical equivalences by substituting $[\varphi_1]$ for the subformula φ_1. Because $\varphi_1 \equiv [\varphi_1] \in \mathcal{C}' = LTL^{\overline{\text{det}}}(AP)$ holds by the induction hypothesis, it is easy to see that $\varphi \equiv [\varphi] \in LTL^{\overline{\text{det}}}(AP)$ holds by the semantics of LTL and the syntactic closure properties of $LTL^{\overline{\text{det}}}(AP)$. The result holds by induction on $|\varphi|$ for all temporal formulas $\varphi \in LTL^{\text{CND}}(AP)$.

<u>$(\mathcal{C} = LTL^{\overline{\text{det}}}(AP), \mathcal{C}' = LTL^{\text{CND}}(AP))$</u> If $\varphi = \big((\theta \vee \varphi_1) \wedge (\neg\theta \vee \varphi_2)\big)$, then it is easy to check from the semantics of LTL that $\varphi \equiv \big((\neg\theta \wedge \varphi_1) \vee (\theta \wedge \varphi_2)\big)$ holds. Similarly, if $\varphi = \big((\theta \vee \varphi_1) \mathsf{R} (\neg\theta \vee \varphi_2)\big)$, then $\varphi \equiv \big(\neg\theta \mathsf{U} ((\neg\theta \wedge \varphi_1) \vee (\theta \wedge \varphi_2))\big)$ holds (see [Maidl 2000b] for a proof). We again obtain $[\varphi]$ by replacing the formulas φ_1 and φ_2 with $[\varphi_1]$ and $[\varphi_2]$ (respectively) in the right-hand side formulas in the identities, and the result follows by the semantics of LTL and the syntactic closure properties of $LTL^{\text{CND}}(AP)$ because $\varphi_1 \equiv [\varphi_1] \in \mathcal{C}'$ and $\varphi_2 \equiv [\varphi_2] \in \mathcal{C}' = LTL^{\text{CND}}(AP)$ hold by the induction hypothesis. We conclude that the result holds for all temporal formulas $\varphi \in LTL^{\overline{\text{det}}}(AP)$ by induction on $|\varphi|$. □

Preservation of the Number of Pure Temporal Subformulas

To show that the mappings $[\,\cdot\,]^{\overline{\text{det}}}$ and $[\,\cdot\,]^{\text{CND}}$ do not increase the number of syntactically distinct pure temporal subformulas, we need the following additional definition. We say that a formula $\varphi \in LTL^{\text{CND}}(AP)$ is in *CND-normal form* iff φ does not contain subformulas of the form $(\varphi_1 \wedge \theta)$ or $(\bot \mathsf{R} \theta)$ (where $\varphi_1 \in LTL^{\text{CND}}(AP) \setminus PL(AP)$, $\theta \in PL(AP)$, and $\mathsf{R} \in \{\mathsf{R_s}, \mathsf{R_w}\}$). Obviously, every formula in $LTL^{\text{CND}}(AP)$ can be rewritten in CND-normal form due to the identities $(\varphi_1 \wedge \theta) \equiv (\theta \wedge \varphi_1)$ and $(\bot \mathsf{R} \theta) \equiv \big(\theta \mathsf{U} (\bot \wedge \theta)\big) \equiv (\theta \mathsf{U} \bot)$ that hold for all $\varphi_1 \in LTL(AP)$ and $\theta \in PL(AP)$ (where the R and U connectives are of the same strength in each formula in the second chain of identities). Furthermore, it is easy to see that the CND-normal form of φ has at most as many pure temporal subformulas as φ itself.

We first make note of the fact that the translation from $LTL^{\text{CND}}(AP)$ to $LTL^{\overline{\text{det}}}(AP)$ maps every pure temporal subformula of φ to a pure temporal subformula of $[\varphi]^{\overline{\text{det}}}$, and the same holds for the other direction.

Lemma 4.6.7 *Let $\mathcal{C}$ be one of the subclasses $LTL^{\text{CND}}(AP)$ or $LTL^{\overline{\text{det}}}(AP)$, let $\mathcal{C}'$ be the opposite subclass, let $\varphi \in \mathcal{C}$, and let $[\varphi]$ be the formula obtained from φ via the translation from $\mathcal{C}$ to $\mathcal{C}'$. For all $\psi \in \mathrm{Temp}(\varphi)$, $[\psi] \in \mathrm{Temp}\big([\varphi]\big)$ holds.*

<u>Proof:</u> The result holds trivially if $\varphi \in PL(AP)$ holds, since φ has no pure temporal subformulas in this case. For temporal formulas, the result follows by a straightforward induction on $|\varphi|$ by observing that for all formulas $\psi \in \mathcal{C}$, ψ is a pure temporal formula only if $[\psi]$ is a pure temporal formula. We omit the details of the proof. □

On the other hand, any two syntactically distinct formulas in CND-normal form in $LTL^{\mathrm{CND}}(AP)$, or two syntactically distinct formulas in $LTL^{\overline{\mathrm{det}}}(AP)$, always translate to syntactically distinct formulas in the opposite subclass.

Lemma 4.6.8 *Let $\mathcal{C}$ be one of the subclasses $LTL^{\mathrm{CND}}(AP)$ or $LTL^{\overline{\mathrm{det}}}(AP)$, let $\mathcal{C}'$ be the opposite subclass, let $\varphi, \psi \in \mathcal{C}$ (and let φ, ψ be in CND-normal form if $\mathcal{C} = LTL^{\mathrm{CND}}(AP)$), and let $[\varphi]$ and $[\psi]$ be the formulas obtained from φ and ψ, respectively, via the translation from $\mathcal{C}$ to $\mathcal{C}'$. If $\varphi \neq \psi$, then $[\varphi]^{\overline{\mathrm{det}}} \neq [\psi]^{\overline{\mathrm{det}}}$.*

<u>*Proof:*</u> Suppose that $\varphi \neq \psi$ holds. We show that $[\varphi]$ and $[\psi]$ are syntactically distinct by induction on the maximum length of φ and ψ. The result is obvious if $\varphi, \psi \in PL(AP)$ holds, since $[\varphi] = \varphi \neq \psi = [\psi]$ holds in this case. In particular, $[\varphi] \neq [\psi]$ holds if $\max\big\{|\varphi|, |\psi|\big\} = 1$. The result follows immediately also if only one of φ and ψ is a temporal formula: for example, if $\mathrm{Temp}(\varphi) \neq \emptyset$ but $\mathrm{Temp}(\psi) = \emptyset$, then $\mathrm{Temp}\big([\varphi]\big) \neq \emptyset$ holds by Lemma 4.6.7, but $\mathrm{Temp}\big([\psi]\big) = \mathrm{Temp}(\psi) = \emptyset$. Obviously, two formulas cannot be syntactically identical if only one of them contains a pure temporal subformula.

Assume that the result holds whenever $\max\big\{|\varphi|, |\psi|\big\} \leq k$ holds for some $1 \leq k < \omega$, and let $\varphi, \psi \in \mathcal{C}$ be temporal formulas with $\max\big\{|\varphi|, |\psi|\big\} = k + 1$.

<u>$(\mathcal{C} = LTL^{\mathrm{CND}}(AP), \mathcal{C}' = LTL^{\overline{\mathrm{det}}}(AP))$</u> Obviously, two compound LTL formulas are syntactically identical only if they have the same main connective. It is easy to check directly from the rewrite rules that $[\varphi]^{\overline{\mathrm{det}}} \neq [\psi]^{\overline{\mathrm{det}}}$ holds if φ and ψ have different main connectives from one of the sets $\{\mathsf{X}, \vee, \wedge, \mathsf{U_s}, \mathsf{U_w}\}$, $\{\mathsf{X}, \vee, \wedge, \mathsf{R_s}, \mathsf{R_w}\}$, $\{\mathsf{U_s}, \mathsf{R_w}\}$ or $\{\mathsf{U_w}, \mathsf{R_s}\}$, because the translation from $LTL^{\mathrm{CND}}(AP)$ to $LTL^{\overline{\mathrm{det}}}(AP)$ either preserves the main connective of every formula (or the strength of the formula's main connective if it is $\mathsf{U_s}$ or $\mathsf{U_w}$). In the remaining case, $\varphi = (\theta \,\mathsf{U}\, \varphi_1)$ and $\psi = (\psi_1 \,\mathsf{R}\, \theta')$ hold for some $\theta, \theta' \in PL(AP)$ and $\varphi_1, \psi_1 \in LTL^{\mathrm{CND}}(AP)$ (and U and R have the same strength).

Suppose that $[\varphi]^{\overline{\mathrm{det}}} = [\psi]^{\overline{\mathrm{det}}}$ holds. From the recursive definition, we see that $[\varphi]^{\overline{\mathrm{det}}} = \big((\neg\theta \vee [\varphi_1]^{\overline{\mathrm{det}}}) \,\mathsf{R}\, (\neg(\neg\theta) \vee [\varphi_1]^{\overline{\mathrm{det}}})\big)$ and $[\psi]^{\overline{\mathrm{det}}} = \big((\neg\theta' \vee [\psi_1]^{\overline{\mathrm{det}}}) \,\mathsf{R}\, (\neg(\neg\theta') \vee \bot)\big)$, and it follows that $\theta = \theta'$, $[\varphi_1]^{\overline{\mathrm{det}}} = [\psi_1]^{\overline{\mathrm{det}}}$ and $[\varphi_1]^{\overline{\mathrm{det}}} = \bot$ must hold in this case. But then also $[\psi_1]^{\overline{\mathrm{det}}} = \bot$ and $\psi_1 = \bot$ hold, and thus we may write $\psi = (\psi_1 \,\mathsf{R}\, \theta') = (\bot \,\mathsf{R}\, \theta')$. This is, however, a contradiction, because ψ is in CND-normal form. Therefore at least one of $\theta \neq \theta'$, $[\varphi_1]^{\overline{\mathrm{det}}} \neq [\psi_1]^{\overline{\mathrm{det}}}$ or $[\varphi_1]^{\overline{\mathrm{det}}} \neq \bot$ holds, and thus $[\varphi]^{\overline{\mathrm{det}}} \neq [\psi]^{\overline{\mathrm{det}}}$.

If φ and ψ have the same main connective, then the result follows easily from the induction hypothesis (in case $\wedge$, use the fact that both φ and ψ are in CND-normal form). We present the case $\mathsf{U_s}$ as an example.

Let $\varphi = (\theta \,\mathsf{U_s}\, \varphi_1)$ and $\psi = (\theta' \,\mathsf{U_s}\, \psi_1)$ for some $\theta, \theta' \in PL(AP)$ and $\varphi_1, \psi_1 \in LTL^{\mathrm{CND}}(AP)$. Because $\varphi \neq \psi$ holds, then either $\theta \neq \theta'$ or $\varphi_1 \neq \psi_1$ holds. In the latter case, $[\varphi_1]^{\overline{\mathrm{det}}} \neq [\psi_1]^{\overline{\mathrm{det}}}$ holds by the induction hypothesis because $\max\big\{|\varphi_1|, |\psi_1|\big\} \leq k$. In both cases, it is easy to see that $[\varphi]^{\overline{\mathrm{det}}} = \big((\neg\theta \vee [\varphi_1]^{\overline{\mathrm{det}}}) \,\mathsf{R_s}\, (\neg(\neg\theta) \vee [\varphi_1]^{\overline{\mathrm{det}}})\big) \neq \big((\neg\theta' \vee [\psi_1]^{\overline{\mathrm{det}}}) \,\mathsf{R_s}\, (\neg(\neg\theta') \vee [\psi_1]^{\overline{\mathrm{det}}})\big) = [\psi]^{\overline{\mathrm{det}}}$, and the result follows.

The result holds by induction for all pairs of syntactically distinct temporal formulas in CND-normal form.

<u>$(\mathcal{C} = LTL^{\overline{\text{det}}}(AP), \mathcal{C}' = LTL^{\text{CND}}(AP))$</u> It is again easy to check from the rewrite rules that $[\varphi]^{\text{CND}} \neq [\psi]^{\text{CND}}$ holds if φ and ψ have different main connectives from one of the sets $\{\mathsf{X}, \vee, \mathsf{R_s}, \mathsf{R_w}\}$ or $\{\mathsf{X}, \wedge, \mathsf{R_s}, \mathsf{R_w}\}$. Suppose that $[\varphi]^{\text{CND}} = [\psi]^{\text{CND}}$ holds for $\varphi = (\varphi_1 \vee \varphi_2)$ and $\psi = \big((\theta \vee \psi_1) \wedge (\neg\theta \vee \psi_2)\big)$ ($\varphi_1, \varphi_2, \psi_1, \psi_2 \in LTL^{\overline{\text{det}}}(AP)$ and $\theta \in PL(AP)$). Then $\big([\varphi_1]^{\text{CND}} \vee [\varphi_2]^{\text{CND}}\big) = \big((\neg\theta \wedge [\psi_1]^{\text{CND}}) \vee (\theta \wedge [\psi_2]^{\text{CND}})\big)$ holds, and thus $[\varphi_1]^{\text{CND}} = \big(\neg\theta \wedge [\psi_1]^{\text{CND}}\big)$ and $[\varphi_2]^{\text{CND}} = \big(\theta \wedge [\psi_2]^{\text{CND}}\big)$ must hold in this case. Because none of the recursive rules for translating a temporal formula from $LTL^{\overline{\text{det}}}(AP)$ to $LTL^{\text{CND}}(AP)$ creates a formula having the operator $\wedge$ as its main connective, however, $[\varphi_1]^{\text{CND}}$ and $[\varphi_2]^{\text{CND}}$ are necessarily propositional formulas, and thus also $\varphi \in PL(AP)$. This contradicts the assumption that both φ and ψ are temporal formulas. It follows that $[\varphi]^{\text{CND}} \neq [\psi]^{\text{CND}}$ holds also in this case.

If φ_1 and φ_2 have the same main connective, then the result follows by the induction hypothesis similarly to the other direction. Therefore, $[\varphi]^{\text{CND}} \neq [\psi]^{\text{CND}}$ holds by induction for all pairs of syntactically distinct temporal formulas $\varphi, \psi \in LTL^{\overline{\text{det}}}(AP)$. □

We can now show that translating formulas between $LTL^{\text{CND}}(AP)$ and $LTL^{\overline{\text{det}}}(AP)$ incurs no blow-up in the number of pure temporal subformulas.

Proposition 4.6.9 *Let $\mathcal{C}$ be one of the syntactic subclasses $LTL^{\text{CND}}(AP)$ or $LTL^{\overline{\text{det}}}(AP)$, let $\mathcal{C}'$ be the opposite subclass, let $\varphi \in \mathcal{C}$, and let $[\varphi]$ be the formula obtained from φ via the translation from $\mathcal{C}$ to $\mathcal{C}'$. The formula $[\varphi]$ has at most as many syntactically distinct pure temporal subformulas as φ.*

<u>*Proof:*</u> If $\mathcal{C} = LTL^{\text{CND}}(AP)$, we assume that φ is in CND-normal form for the inductive proof. The result then follows easily for all formulas $\varphi \in LTL^{\text{CND}}(AP)$, because the CND-normal form of φ has at most as many syntactically distinct pure temporal subformulas as φ.

If φ is a propositional formula, then $[\varphi] = \varphi$ holds, and $\big|\text{Temp}\big([\varphi]\big)\big| = |\text{Temp}(\varphi)| = 0$ obviously holds in this case. In particular, the result holds if $|\varphi| = 1$. Assume that $\big|\text{Temp}\big([\varphi]\big)\big| \leq |\text{Temp}(\varphi)|$ holds for all formulas $\varphi \in \mathcal{C}$ (in CND-normal form, if $\mathcal{C} = LTL^{\text{CND}}(AP)$) with $|\varphi| \leq k$ for some $1 \leq k < \omega$, and let $\varphi \in \mathcal{C}$ be a temporal formula of length $k+1$.

<u>$(\mathcal{C} = LTL^{\text{CND}}(AP), \mathcal{C}' = LTL^{\overline{\text{det}}}(AP))$</u> It is easy to see from the rewrite rules that $\big|\text{Temp}\big([\varphi]\big)\big| \leq |\text{Temp}(\varphi)|$ holds if φ is of the form $\mathsf{X}\varphi_1$, $(\theta \wedge \varphi_1)$, $(\theta \,\mathsf{U}\, \varphi_1)$ or $(\varphi_1 \,\mathsf{R}\, \theta)$ for some $\theta \in PL(AP)$ and $\varphi_1 \in LTL^{\text{CND}}(AP)$, because $\big|\text{Temp}\big([\varphi_1]\big)\big| \leq |\text{Temp}(\varphi_1)|$ holds by the induction hypothesis (clearly $|\varphi_1| \leq k$ holds, and because φ is in CND-normal form, so is φ_1).

If $\varphi = (\varphi_1 \vee \varphi_2)$ (for formulas $\varphi_1, \varphi_2 \in LTL^{\text{CND}}(AP)$ in CND-normal form), then $[\varphi] = \big([\varphi_1] \vee [\varphi_2]\big)$. In this case

$$|\text{Temp}(\varphi)| = |\text{Temp}(\varphi_1)| + |\text{Temp}(\varphi_2)| - |\text{Temp}(\varphi_1) \cap \text{Temp}(\varphi_2)|$$

and

$$\begin{aligned}\big|\text{Temp}\big([\varphi]\big)\big| = \big|\text{Temp}\big([\varphi_1]\big)\big| + \big|\text{Temp}\big([\varphi_2]\big)\big| \\ - \big|\text{Temp}\big([\varphi_1]\big) \cap \text{Temp}\big([\varphi_2]\big)\big|.\end{aligned}$$

Because $|\varphi_1| \leq k$ and $|\varphi_2| \leq k$ hold, $\bigl|\mathrm{Temp}\bigl([\varphi_1]\bigr)\bigr| \leq |\mathrm{Temp}(\varphi_1)|$ and $\bigl|\mathrm{Temp}\bigl([\varphi_2]\bigr)\bigr| \leq |\mathrm{Temp}(\varphi_2)|$ hold by the induction hypothesis. The result obviously follows if also $\bigl|\mathrm{Temp}\bigl([\varphi_1]\bigr) \cap \mathrm{Temp}\bigl([\varphi_2]\bigr)\bigr| \geq |\mathrm{Temp}(\varphi_1) \cap \mathrm{Temp}(\varphi_2)|$ holds. We check that this is indeed the case.

Clearly, if $\mathrm{Temp}(\varphi_1)$ and $\mathrm{Temp}(\varphi_2)$ share a pure temporal subformula $\psi \in \mathrm{Temp}(\varphi_1) \cap \mathrm{Temp}(\varphi_2)$, then $\mathrm{Temp}\bigl([\varphi_1]\bigr)$ and $\mathrm{Temp}\bigl([\varphi_2]\bigr)$ share the pure temporal subformula $[\psi]$ by Lemma 4.6.7. On the other hand, $[\psi] \neq [\psi']$ holds for all pairs of syntactically distinct pure temporal subformulas $\psi, \psi' \in \mathrm{Temp}(\varphi_1) \cap \mathrm{Temp}(\varphi_2)$ $(\psi \neq \psi')$ by Lemma 4.6.8. Therefore, the number of syntactically distinct pure temporal subformulas in $\mathrm{Temp}\bigl([\varphi_1]\bigr) \cap \mathrm{Temp}\bigl([\varphi_2]\bigr)$ cannot be smaller than their number in $\mathrm{Temp}(\varphi_1) \cap \mathrm{Temp}(\varphi_2)$.

The result holds by induction on the length of φ for all temporal formulas $\varphi \in LTL^{\mathrm{CND}}(AP)$ in CND-normal form.

<u>$(\mathcal{C} = LTL^{\overline{\mathrm{det}}}(AP), \mathcal{C}' = LTL^{\mathrm{CND}}(AP))$</u> The proof is analogous to the other direction. The result can be verified directly from the induction hypothesis if $\varphi = \mathsf{X}\varphi_1$ holds for some $\varphi_1 \in LTL^{\overline{\mathrm{det}}}(AP)$, and in the other cases by using Lemma 4.6.7 and Lemma 4.6.8 to establish that $\bigl|\mathrm{Temp}\bigl([\varphi]\bigr)\bigr| \leq |\mathrm{Temp}(\varphi)|$ holds. □

Because the syntactic subclasses $LTL^{\mathrm{CND}}(AP)$ and $LTL^{\overline{\mathrm{det}}}(AP)$ are expressively equivalent, it is possible to combine the definitions of the subclasses to facilitate specifying LTL properties that can be translated directly into self-loop nondeterministic automata. For example, formulas built using the shorter syntactic closure rules of $LTL^{\mathrm{CND}}(AP)$ may be easier to read than the corresponding formulas of $LTL^{\overline{\mathrm{det}}}(AP)$. On the other hand, the closure rules of $LTL^{\overline{\mathrm{det}}}(AP)$ can be used as templates for special translation rules [Maidl 2000a] for building automata from certain LTL formulas.

4.6.5 A Remark on Satisfiability

Because the satisfiability problem for full LTL is **PSPACE**-complete [Sistla and Clarke 1982, 1985], some research effort has been directed at finding subclasses of LTL for which this problem is of lower computational complexity. In particular, this research has resulted in the discovery of several **NP**-complete subclasses of LTL, such as LTL with X or F as the only temporal operator [Sistla and Clarke 1982, 1985], LTL with both of these temporal operators but negation restricted only to atomic formulas [Sistla and Clarke 1982, 1985], subclasses obtained by restricting the number of atomic propositions or the nesting of temporal operators [Demri and Schnoebelen 1998, 2002], and LTL with F and a set of X operators parameterized by the underlying alphabet [Muscholl and Walukiewicz 2004, 2005]. (Of course, **NP**-completeness amounts to lower computational complexity only if **NP** $\neq$ **PSPACE**.)

In this section we show the satisfiability problem in $LTL^{\mathrm{CND}}(AP)$ to be **NP**-complete. Actually, this result follows from the **NP**-completeness of satisfiability in the existential ($\exists$CTL) fragment of the branching time temporal logic CTL [Kupferman and Vardi 1995, 2000]:[5] due to the strict restrictions

[5] Thanks to Orna Kupferman for pointing out this connection.

concerning the closure of $LTL^{\mathrm{CND}}(AP)$ under the $\wedge$, U and R operators, replacing each temporal operator of a formula in $LTL^{\mathrm{CND}}(AP)$ with the corresponding existentially path-quantified CTL operator yields an ∃CTL formula which can be shown to have a nonbranching computation tree model (that is directly identifiable as a word model for the formula in $LTL^{\mathrm{CND}}(AP)$) iff it is satisfiable. We shall nevertheless present an **NP**-completeness proof for satisfiability in the subclass $LTL^{\mathrm{CND}}(AP)$ in detail using a more direct approach since this proof will apply without changes—in contrast to the reduction to ∃CTL satisfiability—also to an extension of the subclass $LTL^{\mathrm{CND}}(AP)$ to be considered later in Sect. 5.6.3.

By Corollary 4.6.4, any formula $\varphi \in LTL^{\mathrm{CND}}(AP) \subseteq LTL(AP)$ can be translated into a self-loop nondeterministic automaton having at most $2 + |\mathrm{Temp}(\varphi)| \leq 1 + |\varphi|$ states. Recall from Sect. 3.2.3 that also the number of acceptance conditions in the automaton is bounded by $|\mathrm{Temp}(\varphi)| \leq |\varphi|$ in this case (because $|\mathrm{Temp}(\varphi)| = \big|\mathrm{Temp}\big([\varphi]^{\mathrm{PNF}}\big)\big|$ holds). These facts imply the following upper bound on the encoding of models of LTL formulas in $LTL^{\mathrm{CND}}(AP)$.

Proposition 4.6.10 *Let $\varphi \in LTL(AP)$ be an LTL formula whose models are fin-recognized by a self-loop nondeterministic automaton (working on the alphabet 2^{AP}) with at most $1 + |\varphi|$ states and $|\varphi|$ acceptance conditions. The formula φ is satisfiable iff there exist finite words $u, v \in (2^{AP})^*$, $|u| \leq |\varphi|$, $1 \leq |v| \leq |\varphi|$, such that $uv^\omega \models \varphi$ holds.*

Proof: (Only if) Assume that φ is satisfiable, i.e., $\mathcal{L}(\varphi) \neq \emptyset$ holds. Let $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ be a nondeterministic automaton that fin-recognizes the language $\mathcal{L}(\varphi)$ (with $|Q| \leq 1 + |\varphi|$ and $\mathcal{F} = \{f_0, f_1, \ldots, f_{n-1}\}$ for some $0 \leq n \leq |\varphi|$), and let $w \in \mathcal{L}(\varphi)$. By the assumption, $\mathcal{A}$ has a fin-accepting run $G = \langle V, E, L\rangle$ on w. Because $\mathcal{A}$ is nondeterministic, Δ contains no transitions with an empty set of target states, and thus G contains an infinite branch $\beta \in \mathcal{B}(G)$ (L is consistent).

Because $\mathcal{A}$ is a self-loop nondeterministic automaton, the branch β converges to a nontransient state $q \in Q$ of $\mathcal{A}$ by Corollary 2.3.19. It is easy to see that we can extract from β a chain of edges $e_0, e_1, \ldots, e_k \in E$ (for some $0 \leq k < \omega$) such that the labels of the source nodes of these edges form a simple path $(q_i)_{0 \leq i \leq k}$ from q_I to q for some $0 \leq k \leq |Q| - 1 \leq 1 + |\varphi| - 1 = |\varphi|$. Because G is a run, the labels of the edges e_i form a chain of transitions $(t_i)_{0 \leq i \leq k} \in \Delta^{k+1}$, where $t_i = \langle q_i, \Gamma_i, F_i, Q_i'\rangle$ (with $\Gamma_i \neq \emptyset$ and $Q_i' = \{q_{i+1}\}$) holds for all $0 \leq i < k$, and $Q_{k-1}' = \{q\}$. Additionally, because q is a nontransient state of the nondeterministic automaton $\mathcal{A}$, there exists for all $0 \leq j < n' \stackrel{\mathit{def}}{=} \max\{n, 1\}$ a transition $t_j' = \big\langle q, \Gamma_j', F_j', \{q\}\big\rangle \in \Delta$ with $\Gamma_j' \neq \emptyset$ and (if $j < n$) $f_j \notin F_j$.

Let $\sigma_i \in \Gamma_i$ and $\sigma_j' \in \Gamma_j'$ for all $0 \leq i < k$ and $0 \leq j < n'$, respectively, let $u \stackrel{\mathit{def}}{=} \sigma_0\sigma_1 \cdots \sigma_{k-1}$, and let $v \stackrel{\mathit{def}}{=} \sigma_0'\sigma_1' \cdots \sigma_{n'-1}'$. Clearly, $|u| = k \leq |\varphi|$ and $1 \leq |v| = n' = \max\{n, 1\} \leq |\varphi|$ hold. We show that $\mathcal{A}$ has a fin-accepting run on the word $w' \stackrel{\mathit{def}}{=} uv^\omega$. Let $G' = \langle V', E', L'\rangle$, where

- $V' \stackrel{\mathit{def}}{=} \bigcup_{0 \leq i < \omega} V_i'$, and $V_i' \stackrel{\mathit{def}}{=} \{v_i\}$ for all $0 \leq i < \omega$;
- $E' \stackrel{\mathit{def}}{=} \bigcup_{0 \leq i < \omega} \big\{\langle v_i, V_{i+1}'\rangle\big\}$;

- $L'(v_i) \stackrel{def}{=} q_i$ and $L'\big(\langle v_i, V'_{i+1}\rangle\big) \stackrel{def}{=} t_i$ for all $0 \leq i < k$, and $L'(v_i) \stackrel{def}{=} q$ and $L'\big(\langle v_i, V'_{i+1}\rangle\big) \stackrel{def}{=} t'_{(i-k) \bmod |v|}$ for all $k \leq i < \omega$.

Obviously, G' satisfies the partitioning and causality constraints, and $L'(v_0) = q_0 = q_I$. Let $e = \langle v_i, V'_{i+1}\rangle \in E'$ be an edge in G' for some $0 \leq i < \omega$. If $i < k$, then $L'(e) = t_i = \langle q_i, \Gamma_i, F_i, Q'_i\rangle = \big\langle L'(v_i), \Gamma_i, F_i, L'(V'_{i+1})\big\rangle \in \Delta$, and because $w'(i) = u(i) = \sigma_i \in \Gamma_i$, L is consistent in this case. Otherwise $L'(e) = t'_{(i-k) \bmod |v|} = \big\langle q, \Gamma'_{(i-k) \bmod |v|}, F'_{(i-k) \bmod |v|}, \{q\}\big\rangle = \big\langle L'(v_i), \Gamma'_{(i-k) \bmod |v|}, F'_{(i-k) \bmod |v|}, L'(V'_{i+1})\big\rangle \in \Delta$, and L is consistent, because $w'(i) = v\big((i-k) \bmod |v|\big) = \sigma'_{(i-k) \bmod |v|} \in \Gamma'_{(i-k) \bmod |v|}$ holds. It is easy to see that G' contains a unique infinite branch, and this branch is fin-accepting, because for all $0 \leq i < n$, $f_i \notin F_j$ holds for all $k \leq j < \omega$ such that $i = (j-k) \bmod |v|$. Thus $\mathcal{A}$ fin-accepts uv^ω, and by Theorem 3.4.2, $uv^\omega \models \varphi$ holds.

(If) If $uv^\omega \models \varphi$ holds, then $uv^\omega \in \mathcal{L}(\varphi) \neq \emptyset$, and thus φ is satisfiable. $\square$

Proposition 4.6.10 leads to the following result on computational complexity.

Corollary 4.6.11 *Let AP be a countably infinite set of atomic propositions. The satisfiability problem for $LTL^{\mathrm{CND}}(AP)$ is* $\mathbf{NP}$*-complete.*

Proof: Because the satisfiability problem for formulas in $PL(AP)$ that are in conjunctive normal form is $\mathbf{NP}$-hard [Cook 1971], $\mathbf{NP}$-hardness follows trivially because $PL(AP) \subseteq LTL^{\mathrm{CND}}(AP)$. Let $\varphi \in LTL^{\mathrm{CND}}(AP)$. By Proposition 4.6.10, φ is satisfiable iff there exist words $u, v \in (2^{AP})^*$ of length at most $|\varphi|$ such that $uv^\omega \models \varphi$ holds. A nondeterministic decision procedure for testing the satisfiability of φ guesses two words $u, v \in (2^{AP})^*$ of length at most $|\varphi|$ and checks whether $uv^\omega \models \varphi$ holds. This check can be done in polynomial time in $\big(|u| + |v|\big) \cdot |\varphi| \in O\big(|\varphi|^2\big)$ [Wolper 1987] (for example, using the algorithm of Clarke et al. [1983, 1986]). The satisfiability of φ can now be decided in nondeterministic polynomial time because also the words u and v can be constructed in polynomial time in $|\varphi|$. (When choosing a subset of AP, it is sufficient to restrict to those propositions which occur in φ; because also the number of these propositions is bounded by $|\varphi|$, u and v can be constructed in polynomial time in $|\varphi|$.) $\square$

5 REFINING THE BASIC TRANSLATION RULES

In this chapter we explore methods for improving the basic translation procedure from LTL into self-loop alternating automata by refining the translation rules defined in Sect. 3.1. Except for mentioning some standard syntactic heuristics for reducing the number of temporal subformulas in an LTL formula and for representing and simplifying guards of transitions in automata built using the translation rules (Sect. 5.1), we take a high-level theoretical approach to optimizing the translation rules by exploiting language containment relationships between self-loop alternating automata (Sect. 5.2). In addition to using language containment tests between automata built in the translation as simple “pre- and postprocessing” steps for the translation rules (Sect. 5.3), we shall also refine the translation rules themselves (Sect. 5.4, Sect. 5.5): instead of building a compound automaton from one or two component automata always in the same way, taking the semantic properties of the component automata into account may make it possible to construct compound automata with a simpler transition structure than the one determined by the basic rules. Some of the refined rules even prove to be universally applicable. Finally, we compare the new translation rules with the original ones and use them to extend the class of LTL formulas that can be translated into nondeterministic automata without applying the universal subset construction (Sect. 5.6).

5.1 SIMPLE OPTIMIZATIONS

5.1.1 Subformulas with Commutative Main Connectives

A standard basic heuristic for reducing the number of steps required for translating (the positive normal form of) a formula $\varphi \in LTL(AP)$ into an automaton is to order the top-level subformulas of each subformula of $[\varphi]^{\mathrm{PNF}}$ with a commutative binary main connective ($\vee$ or $\wedge$ in our set of basic operators) systematically (e.g., using a lexicographic ordering of LTL formulas), for example, as an explicit preprocessing step before the translation. Even though fixing the order of subformulas has no effect on the number of states in the resulting automaton (by Corollary 3.2.2, this number does not depend on the subformulas of $[\varphi]^{\mathrm{PNF}}$ with a Boolean main connective), it may nevertheless decrease the total number of translation steps, which is proportional to the number of syntactically distinct subformulas in φ. This way, the translation procedure may also avoid repeating potentially expensive automaton minimization operations (such as those presented later in this chapter) on automata built for logically equivalent subformulas of $[\varphi]^{\mathrm{PNF}}$ that differ only in the order of their top-level subformulas.

5.1.2 Transition Guard Simplification

As noted in Sect. 3.1.1, all transition guards of an automaton constructed using the translation rules are finite conjunctions of one or more atomic formulas (also referred to as *terms* in the literature [Etessami and Holzmann 2000]). Due to the associativity of the $\wedge$ operator, each guard formula $\theta \in PL(AP)$ can be expressed (omitting parentheses) in the form $\theta = \mu_1 \wedge \mu_2 \wedge \cdots \wedge \mu_n$ for some $1 \leq n < \omega$, where $\mu_i \in PL(AP)$ is an atomic formula for all $1 \leq i \leq n$. The formula can then be simplified using the commutativity of $\wedge$ and the classic identities of propositional logic:

$$\begin{array}{cc} (\theta \wedge \theta) \equiv \theta & (\theta \wedge \neg\theta) \equiv (\neg\theta \wedge \theta) \equiv \bot \\ (\theta \wedge \top) \equiv (\top \wedge \theta) \equiv \theta & (\theta \wedge \bot) \equiv (\bot \wedge \theta) \equiv \bot \end{array}$$

It is easy to see that every guard formula can thus be written in a form in which it is either equal to one of the Boolean constants, or it is a conjunction of literals formed from distinct atomic propositions. Because the formula $\bot$ is unsatisfiable (i.e., because $\mathcal{L}(\bot) = \emptyset$ holds), it follows immediately by Corollary 2.3.11 that all transitions with $\bot$ as a guard can be removed from the alternating automaton without changing its language.

Several authors have suggested using binary decision diagrams (BDDs) [Bryant 1986] for encoding and manipulating transition guards expressed as Boolean formulas [Couvreur 1999; Thirioux 2002; Latvala 2003]. However, as observed by Gastin and Oddoux [2001], guards given in the above very restricted form (a conjunction of literals) can be encoded and manipulated efficiently as pairs of sets of atomic propositions. More precisely, the conjunction of literals $\theta = \mu_1 \wedge \mu_2 \wedge \cdots \wedge \mu_n \in PL(AP)$ can be encoded as the pair $\langle P_1, P_2 \rangle \in 2^{AP} \times 2^{AP}$, where P_1 (P_2) collects all atomic propositions $p \in AP$ for which $\mu_i = p$ ($\mu_i = \neg p$) holds for some $1 \leq i \leq n$; in this notation, the Boolean constant $\top$ corresponds to the pair $\langle \emptyset, \emptyset \rangle$. This encoding allows straightforward implicit application of the above propositional identities when building new guards from previously defined ones during the translation. It is also easy to check for the validity of propositional implications between two transition guards (a prerequisite for many automaton minimization operations presented in this chapter and in the literature) without using the full power of BDDs. More precisely, if $\langle P_1, P_2 \rangle$ and $\langle P_1', P_2' \rangle$ are the encodings of the guard formulas $\theta \in PL(AP)$ and $\theta' \in PL(AP)$, respectively, then $(\theta \wedge \theta')$ is represented by the pair $\langle P_1 \cup P_1', P_2 \cup P_2' \rangle$, θ is unsatisfiable iff $P_1 \cap P_2 \neq \emptyset$ holds, and the propositional implication $(\theta \rightarrow \theta')$ is valid iff $P_1' \subseteq P_1$ and $P_2' \subseteq P_2$ hold.

Example 5.1.1 Using the above simplification rules for transition guards, the automaton built in Ex. 3.1.1 for the LTL formula $\Big(\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big) \wedge \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big)\Big)$ (repeated in Fig. 5.1 (a)) can be replaced with the one shown in Fig. 5.1 (b) before using the automaton as a component in another translation rule. ■

5.2 LANGUAGE CONTAINMENT CHECKING BETWEEN SELF-LOOP ALTERNATING AUTOMATA

Informally, techniques for the optimization and minimization of automata are used to transform automata into "simpler" automata such that the op-

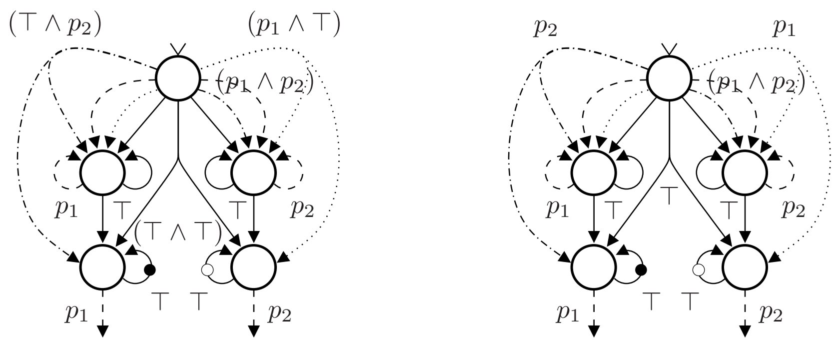


Fig. 5.1: Simplifying the transition guards of an automaton. (a) An automaton built for the LTL formula $\Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big) \wedge \big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big)$; (b) Automaton obtained from (a) via transition guard simplification

timized automata can be substituted for the original automata in all possible applications. Although the formal meaning of "simple" may vary from context to context, this assumption on substitutability mandates that an optimized automaton should, at the very least, recognize the same language as the original one. It is therefore not surprising that constraints needed to ensure that techniques designed for optimizing the automaton have this property are often naturally expressible in a unified way in terms of *language equivalence* (or weaker *language containment*) relationships between automata. We shall give examples of such conditions applied to the optimization of self-loop alternating automata throughout this and the following chapter. Before discussing these applications, however, we review the basic theoretical principles of language containment testing in this section and point out observations specific to handling this task with self-loop alternating automata.

Questions on language containment are traditionally answered computationally by reusing effective decision procedures for checking language emptiness. Given two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ of infinite words over a finite alphabet Σ, the classic reformulation of the language containment relationship $\mathcal{L}_1 \subseteq \mathcal{L}_2$ as language emptiness follows directly from the observation that all words in the language $\mathcal{L}_1$ belong to the language $\mathcal{L}_2$ iff no word in $\mathcal{L}_1$ belongs to the complement of $\mathcal{L}_2$ with respect to Σ^ω, i.e., iff the set intersection $\mathcal{L}_1 \cap \overline{\mathcal{L}_2}$ is empty. If the languages $\mathcal{L}_1$ and $\mathcal{L}_2$ correspond to languages $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)$ and $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$ (fin-)recognized by two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ (respectively) from a class of automata which is closed under operations for constructing automata for Boolean combinations of languages, it follows that the emptiness of the language $\mathcal{L}_1 \cap \overline{\mathcal{L}_2} = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1) \cap \overline{\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)}$ can be decided effectively by applying a decision procedure for language emptiness to an automaton built from $\mathcal{A}_1$ and $\mathcal{A}_2$ using these operations. In particular, the complement language $\overline{\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)}$ is recognized by an automaton built from the automaton $\mathcal{A}_2$ using a complementation procedure.

The powerful result of Muller and Schupp [1985, 1987] provides an elegant theoretical construction for complementing an alternating automaton by "dualizing" its transition relation and its acceptance mode (for formal definitions, see [Muller and Schupp 1985, 1987]). The apparent simplicity of the construction is due, however, to a definition of alternating automata which uses a symbolic Boolean encoding for the transition relation of the automaton, but at the same time (in effect) restricts the guards of transitions to individual elements of the automaton's alphabet. These requirements are not directly compatible with our conventions for representing alternating automata having the alphabet 2^{AP}. On the one hand, the individual transitions of the automaton are not always easily distinguishable from the symbolic Boolean encoding of the transition relation of the automaton; on the other hand, however, allowing Boolean formulas as guards of transitions (Sect. 3.1) makes it possible to represent many transitions on individual symbols of the automaton's alphabet as a single transition if the transitions share their target states and acceptance conditions. A further practical difficulty is caused by the modification to the acceptance mode of the automaton in the complementation construction. Because Boolean operations on automata are usually designed for automata working in the same acceptance mode, it is in our case not possible to combine the automaton built from the automaton $\mathcal{A}_2$ via dualization directly with the automaton $\mathcal{A}_1$ to obtain an automaton for the language $\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \overline{\mathcal{L}_{\text{fin}}(\mathcal{A}_2)}$. Although the acceptance mode of the dualized automaton can be easily reduced back to the original one when working with (very) weak alternating automata (whose acceptance mode can be identified with a special case of state-based inf- or fin-acceptance using a single acceptance condition), this reduction does not generalize directly into self-loop automata with multiple acceptance conditions associated with transitions. This last difficulty could perhaps be overcome by replacing the acceptance conditions in the automaton $\mathcal{A}_2$ with a single condition before dualization—for example, by designing a generalized version of one of the constructions used for this purpose with nondeterministic automata [Emerson and Sistla 1984a,b; Courcoubetis et al. 1991, 1992]—and mapping transition-based acceptance into state-based acceptance (which could be combined with the reduction in the number of acceptance conditions; see, for example, [Gastin and Oddoux 2001]). The resulting automaton could then be complemented via an intermediate translation to weak alternating automata as suggested by Kupferman and Vardi [1997, 2001] (see also [Löding 1998; Thomas 1999] for a closely related approach).

It is apparent from the above discussion that the direct adaptation of complementation constructions from the literature to our definition of alternating automata depends on many intricate intermediate automata transformations, all of which add to the challenge of implementing a complementation procedure correctly. For self-loop alternating automata, however, a general complementation construction is not necessarily needed because of the expressive equivalence between these automata and propositional linear time temporal logic. If the self-loop alternating automata $\mathcal{A}_1$ and $\mathcal{A}_2$ recognize the languages of two LTL formulas φ_1 and φ_2, respectively, it follows directly from the correctness of the basic translation procedure between LTL and self-loop alternating automata (Theorem 3.3.2) that the language $\overline{\mathcal{L}_{\text{fin}}(\mathcal{A}_2)}$ can be fin-recognized by an automaton built from the formula $[\neg\varphi_2]^{\text{PNF}}$. Knowledge of the LTL formulas corresponding to the given self-loop automata thus makes it possible to avoid using a direct complementation construction for automata by reusing the translation procedure from LTL into automata.

Because the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ used in language containment checks to be discussed in this chapter will often have been obtained in substeps in the translation of an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ into an automaton, the correspondence between the automata built in the translation and the node subformulas of φ ensures that the formula $[\neg\varphi_2]^{\mathrm{PNF}}$ is already known—or can be determined easily by syntactic techniques—when testing whether $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1) \cap \overline{\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)} = \emptyset$ (equivalently, whether $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1) \subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$) holds. In this case the automaton built for the language $\overline{\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)}$ also shares its acceptance mode with the automaton $\mathcal{A}_1$ by construction and can therefore be directly combined with this automaton in a decision procedure for language emptiness. Additionally, because $\bigl|\mathrm{Temp}\bigl([\varphi]^{\mathrm{PNF}}\bigr)\bigr| = \bigl|\mathrm{Temp}\bigl([\neg\varphi]^{\mathrm{PNF}}\bigr)\bigr|$ holds for all LTL formulas $\varphi \in LTL(AP)$, the automaton built from the formula $[\neg\varphi_2]^{\mathrm{PNF}}$ has at most $1 + \bigl|\mathrm{Temp}\bigl([\varphi_2]^{\mathrm{PNF}}\bigr)\bigr|$ states by Corollary 3.2.2. If the automaton $\mathcal{A}_2$ was originally built from the formula φ_2, the automaton for the complement language has at most as many states as the automaton $\mathcal{A}_2$, identically to an automaton built from $\mathcal{A}_2$ by dualization. However, all means of complementing self-loop alternating automata conforming to our definition will in the worst case cause an exponential blow-up in the number of transitions in the automaton due to the explicit representation of transitions (for example, when complementing automata built from Boolean formulas in disjunctive normal form with two or more literals in each disjunct).

Nevertheless, it is not always directly possible to reuse the basic translation procedure from LTL into automata to handle language containment tests if the LTL formulas corresponding to the automata $\mathcal{A}_1$ or $\mathcal{A}_2$ are not known beforehand in the context. In this case, direct dualization can still be avoided by applying the reverse translation discussed in Sect. 3.4 to, for example, the automaton $\mathcal{A}_2$ to find an LTL formula to be negated and translated back into an automaton for the language $\overline{\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)}$. In this work, however, we shall focus mostly on special cases that can be handled without applying reverse translation. Even though giving up the ability to complement arbitrary self-loop alternating automata (i.e., the ability to perform arbitrary language containment tests) will reduce the opportunities for the simplification of automata, we shall obtain a collection of optimizations in which all language containment assumptions can be checked by applying the basic translation procedure to LTL formulas that are easily determined in the context.

Finally, it needs be pointed out that the power of language containment testing comes at a cost of high computational complexity, and thus optimizations based on it are unlikely to scale well in practice to large problem instances. Clearly, checking for language containment between automata built from two subformulas of an LTL formula to be translated into an automaton is essentially equivalent to checking for language containment between the formulas themselves; however, this latter problem is easily seen to be already **PSPACE**-hard in the length of the formulas since the decision problem for the satisfiability of LTL formulas is **PSPACE**-complete [Sistla and Clarke 1982, 1985]. (Obviously, an LTL formula φ is not satisfiable iff $\mathcal{L}(\varphi) \subseteq \mathcal{L}(\bot)$ holds.) As already seen in Sect. 3.2.2 and Sect. 3.4, our constructions for building automata from LTL formulas and vice versa are inherently exponential in the length of formulas and the number of states in the automata.

Table 5.1: LTL equivalences under various language containment relationships

Formula	Relationship between languages: $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$	$\mathcal{L}(\varphi_1) \subseteq \overline{\mathcal{L}(\varphi_2)}$	$\overline{\mathcal{L}(\varphi_1)} \subseteq \mathcal{L}(\varphi_2)$	$\mathcal{L}(\varphi_2) \subseteq \mathcal{L}(\varphi_1)$
$(\varphi_1 \vee \varphi_2)$	φ_2		$\top$	φ_1
$(\varphi_1 \wedge \varphi_2)$	φ_1	$\bot$		φ_2
$(\varphi_1 \,\mathsf{U}_\mathsf{s}\, \varphi_2)$	φ_2		$(\top \,\mathsf{U}_\mathsf{s}\, \varphi_2)$	$(\varphi_2 \,\mathsf{R}_\mathsf{s}\, \varphi_1)^\dagger$
$(\varphi_1 \,\mathsf{U}_\mathsf{w}\, \varphi_2)$	φ_2		$\top$	$(\varphi_2 \,\mathsf{R}_\mathsf{w}\, \varphi_1)^\dagger$
$(\varphi_1 \,\mathsf{R}_\mathsf{s}\, \varphi_2)$	$(\varphi_2 \,\mathsf{U}_\mathsf{s}\, \varphi_1)^\dagger$	$\bot$		φ_2
$(\varphi_1 \,\mathsf{R}_\mathsf{w}\, \varphi_2)$	$(\varphi_2 \,\mathsf{U}_\mathsf{w}\, \varphi_1)^\dagger$	$(\bot \,\mathsf{R}_\mathsf{w}\, \varphi_2)$		φ_2

† See also Sect. 5.5.

5.3 RULE PREPROCESSING USING LANGUAGE CONTAINMENT

A language containment relationship between automata built for the top-level subformulas of a binary LTL formula (equivalently, between the subformulas themselves) may allow simplifying the compound automaton built from these automata. For example, if the language of an LTL formula $\varphi_1 \in LTL(AP)$ is contained in the language of another formula $\varphi_2 \in LTL(AP)$ (i.e., if $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$ holds), then $\mathcal{L}\big((\varphi_1 \wedge \varphi_2)\big) = \mathcal{L}(\varphi_1) \cap \mathcal{L}(\varphi_2) = \mathcal{L}(\varphi_1)$ holds. Therefore the automaton already built for φ_1 can be reused to obtain an automaton for the formula $(\varphi_1 \wedge \varphi_2)$ directly, instead of defining the automaton by applying the translation rule given for the $\wedge$ connective. Reusing the automaton built for φ_1 will thus likely result in a smaller automaton for the compound formula than the one that would have been obtained by using the translation rule: for example, there is no need to take a new initial state for the automaton for the compound formula.

Table 5.1 contains formula simplification rules that follow from a number of possible assumptions about the relationship between the languages of two LTL formulas. Each cell in the table gives an LTL formula that is logically equivalent to the formula labeling the row of the cell under the language containment relationship given as the label of the cell's column; empty cells correspond to cases in which the assumptions do not lead to direct simplification of the formulas. The correctness of each rule can be verified directly from the semantics of LTL. Because the LTL formulas referred to in the conditions for language containment are subformulas of the compound formulas labeling the rows of the table, the complement of the language of each of these subformulas (required in the language containment test) can be recognized by an automaton built directly for the positive normal form of the negated subformula.

The expressive equivalence between LTL formulas and self-loop alternating automata guarantees that an automaton built for a formula in a nonempty cell of Table 5.1 can always be substituted for the compound automaton built for the formula labeling the row of the cell. The formula substitution is clearly preferable whenever either one of the original formula's subformulas can be discarded (or replaced with a Boolean constant) in the substitution. However, under some assumptions on language containment, the replacement formula for an Until (Release) formula is a Release (Until) formula of the corresponding strength with only the order of the subformulas reversed. We shall investigate these cases further in Sect. 5.5, where we use the same language containment assumptions to refine the translation rules for these connectives. The identities in Table 5.1 can nevertheless be used for reducing the effective number of distinct subformulas to which the translation needs to be applied when translating a formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ into an automaton. For example, by treating all subformulas of the form $(\varphi_1 \,\mathsf{R_s}\, \varphi_2) \in \mathrm{Sub}(\varphi)$, where $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$, systematically as formulas of the form $(\varphi_2 \,\mathsf{U_s}\, \varphi_1)$, an automaton built for the formula $(\varphi_2 \,\mathsf{U_s}\, \varphi_1)$ can obviously be reused as the automaton for the formula $(\varphi_1 \,\mathsf{R_s}\, \varphi_2)$. This substitution effectively reduces the number of syntactically distinct temporal subformulas in φ and therefore also the worst-case size of an automaton for φ (Corollary 3.2.2). Furthermore, the basic translation rules suggest that it may be preferable to replace Release formulas with Until formulas and not vice versa, since the worst-case number of initial transitions in a compound automaton created by a Release rule is proportional to the product instead of the sum of the numbers of the initial transitions in the component automata.

The simplification rules in Table 5.1 reduce to well-known easy-to-check special cases when one of the subformulas involved in the language containment test is a Boolean constant, due to the fact that

$$\emptyset = \mathcal{L}(\bot) = \overline{\mathcal{L}(\top)} \subseteq \mathcal{L}(\varphi) \subseteq \mathcal{L}(\top) = \overline{\mathcal{L}(\bot)} = (2^{AP})^{\omega}$$

holds for all LTL formulas $\varphi \in LTL(AP)$. These special cases, together with the identities $\mathsf{X}\top \equiv \top$ and $\mathsf{X}\bot \equiv \bot$ for the Next Time connective, can be checked syntactically from the formulas and can therefore be used even if checking for general language containment is considered too expensive. Checking for syntactic special cases that imply language containment is a standard technique used in most actual implementations (for example, [Etessami and Holzmann 2000; Somenzi and Bloem 2000; Thirioux 2002]) usually as a preprocessing step; clearly, performing the translation incrementally supports combining the formula rewriting step easily with the translation itself.

In addition to checking for language containment relationships before applying a translation rule, language containment checks can also be used immediately after applying the rule to test whether the language accepted by the newly defined automaton for a compound formula φ is empty ($\mathcal{L}(\varphi) \subseteq \mathcal{L}(\bot)$), universal (that is, equal to $(2^{AP})^{\omega}$, $\mathcal{L}(\top) \subseteq \mathcal{L}(\varphi)$), or equal to the language of another LTL formula φ' corresponding to some previously built automaton ($\mathcal{L}(\varphi) \subseteq \mathcal{L}(\varphi')$ and $\mathcal{L}(\varphi') \subseteq \mathcal{L}(\varphi)$). Obviously, such relationships may again allow reducing the compound automaton into a simpler one by, for example, improving the opportunities for Boolean constant propagation at the cost of an increase in the number of language containment tests.

5.4 THE $\wedge$ CONNECTIVE

In this section we define a new translation rule for the $\wedge$ connective that can be used instead of the basic one to improve the translation in some simple special cases. We start with a discussion to illustrate opportunities for optimizing the translation rules.

Fig. 5.2: Two automata for the LTL formula $(\mathsf{G}p_1 \wedge \mathsf{G}p_2)$. (a) Automaton built for the formula using the basic translation rules; (b) Minimal automaton for the formula

Intuitively, the basic translation rule for the $\wedge$ connective defines the initial transitions of the automaton for an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ with a subformula of the form $(\varphi_1 \wedge \varphi_2) \in \mathrm{Sub}(\varphi)$ by merging all pairs of initial transitions of the component automata built for the formulas φ_1 and φ_2, "unrolling" all initial self-loops of the automata (see Fig. 3.1 (f), p. 43). Because the initial state of every component automaton with an initial self-loop thus becomes reachable from the initial state of the automaton built for the formula $(\varphi_1 \wedge \varphi_2)$, the state will remain reachable also from the initial state of the automaton obtained for the formula φ at the end of the translation. It is easy to see that this feature of the translation rule leads to the construction of automata with a suboptimal number of states and transitions already in very simple cases.

Example 5.4.1 Consider the translation of the LTL formula

$$(\mathsf{G}p_1 \wedge \mathsf{G}p_2) \equiv \big((\bot \,\mathsf{R_s}\, p_1) \wedge (\bot \,\mathsf{R_s}\, p_2)\big) \in LTL\big(\{p_1, p_2\}\big)$$

into a self-loop alternating automaton using the basic translation rules. Applying the basic translation rule for the $\wedge$ connective to the automata built for the formulas $(\bot \,\mathsf{R_s}\, p_1)$ and $(\bot \,\mathsf{R_s}\, p_2)$ results in the three-state automaton shown in Fig. 5.2 (a). Obviously, this automaton is not of minimal size in the number of states or transitions, since the language of the LTL formula $(\mathsf{G}p_1 \wedge \mathsf{G}p_2)$ is clearly fin-recognized also by the single-state automaton shown in Fig. 5.2 (b). ■

Obviously, an automaton built using the basic translation rules could be optimized afterwards to reduce the number of states in the automaton. In the following, however, we explore techniques for refining the translation rules themselves to make them handle simple optimizations (such as the one shown in Ex. 5.4.1) directly.

Consider two self-loop alternating automata $\mathcal{A}_1$ and $\mathcal{A}_2$, both of which fin-accept a word $w \in \Sigma^\omega$ over their common alphabet Σ. By Proposition 2.3.7, each of these runs begins with a (possibly empty) chain of edges labeled with initial self-loops of the respective automaton. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are used as components in an application of the translation rule given for the $\wedge$ connective, then the automaton $\mathcal{A}$ built using the rule simulates the synchronous behavior of $\mathcal{A}_1$ and $\mathcal{A}_2$ by, in effect, using its initial transitions to spawn copies of $\mathcal{A}_1$ and $\mathcal{A}_2$ (or their subautomata), which then work in parallel on the rest of the input as discussed in Sect. 3.1.1. Consequently, every run of $\mathcal{A}$ separates into

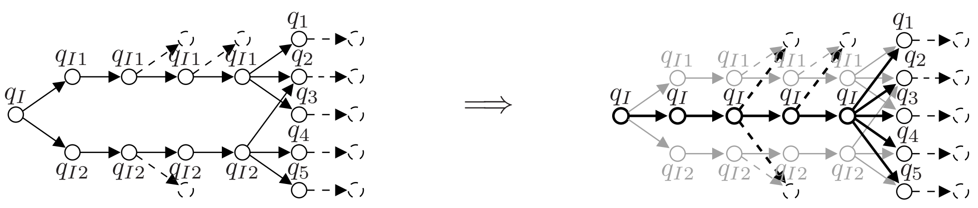


Fig. 5.3: Merging the initial segments of two branches in a run of an automaton having initial state q_I

two (not necessarily disjoint) partitions corresponding to fin-accepting runs of $\mathcal{A}_1$ and $\mathcal{A}_2$. If both of these runs begin with nonempty chains of edges labeled with initial self-loops of $\mathcal{A}_1$ and $\mathcal{A}_2$, the separation of the run of $\mathcal{A}$ into two partitions could be postponed by merging initial segments of these chains together as shown in Fig. 5.3. Furthermore, if no copy of $\mathcal{A}_1$ ($\mathcal{A}_2$) remains in the initial state of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) after the separation, the compound automaton $\mathcal{A}$ can avoid spawning subautomata rooted at the initial states of these component automata. If all runs of $\mathcal{A}$ admit a similar modification, $\mathcal{A}$ does not need any initial transitions to one or both of the initial states of the automata $\mathcal{A}_1$ or $\mathcal{A}_2$. Unlike in the basic construction, the initial state of at least one component automaton now remains a possible candidate for eventual removal from the automaton obtained at the end of the translation (provided, of course, that no subsequent application of a translation rule creates transitions to this state, either).

The above principle of merging the initial segments of two branches in a run of a compound automaton $\mathcal{A}$ can be made implicit in the translation by modifying the translation rule for the $\wedge$ connective. Similarly to the basic construction, we first collect all pairs of initial transitions of the component automata and use the original construction to merge all pairs of transitions, one of which is not a self-loop. However, each pair of self-loops is now tested whether it could be merged into an initial self-loop of $\mathcal{A}$ that simulates the pair of transitions. Since $\mathcal{A}$ should still be allowed to take a transition iff the component automata can take a pair of synchronous transitions, it is again natural to form the guard of the self-loop as the logical conjunction of the guards of the individual transitions. The target states of each initial transition of $\mathcal{A}$ would normally be formed as the union of the target states of the transitions to be merged; however, since we intend to prevent the run of $\mathcal{A}$ from separating into two partitions, we replace the initial states of the component automata in this set with the initial state of $\mathcal{A}$. To define the acceptance conditions of the new self-loop, it is tempting to try a similar strategy by taking the union of the acceptance conditions of the pair of transitions to be merged to prevent $\mathcal{A}$ from possibly fin-recognizing more inputs than its component automata. Analogously to designing a universal subset construction for translating a self-loop alternating automaton into a nondeterministic one (Sect. 4.2.1), this intuitive idea is, however, incorrect in the general case.

Example 5.4.2 Figure 5.4 depicts two alternating automata working on the alphabet $2^{\{p_1,p_2\}}$, together with a self-loop alternating automaton obtained by the above informal construction for merging initial self-loops of the two automata into initial self-loops of the third automaton. It is easy to see that

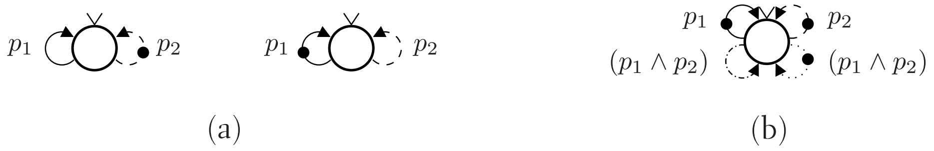


Fig. 5.4: Self-loops of automata that share common acceptance conditions cannot always be merged into a single self-loop by forming their acceptance conditions as the union of the acceptance conditions of the self-loops. (a) Two automata sharing a common acceptance condition; (b) Automaton obtained from the two automata by merging pairs of their initial transitions (taking unions of their acceptance conditions)

both automata in Fig. 5.4 (a) fin-accept the word $\big(\{p_1\}\{p_2\}\big)^\omega$, but this word does not belong to the language of the automaton in Fig. 5.4 (b): on this input, this automaton can take only $\bullet$-transitions, and thus the run of this automaton on this input is not fin-accepting. ■

Evidently, the suggested construction for merging initial self-loops of two component automata is not correct in the general case without a means to distinguish between the conditions inherited from the component automata in the compound automaton. As in the universal subset construction of Sect. 4.2.1, this problem could perhaps be overcome by introducing new acceptance conditions. We shall not explore this approach further in this section, however; instead, we show that the suggested construction is actually correct for a restricted class of automata that includes, in particular, all automata built from LTL formulas using the basic translation rules. The rest of this section is devoted to showing this result. (Actually, the result shown below is slightly more general: instead of merging only pairs of initial self-loops of two component automata into self-loops of a compound automaton, it is permissible to merge any two initial transitions of the component automata if the union of their target states covers their initial states.) The result leads to a new translation rule for the $\land$ connective (Table 5.2); see also Fig. 5.5.

Proposition 5.4.3 *Let $\mathcal{A}_1 = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$ and $\mathcal{A}_2 = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2\rangle$ be two alternating automata ($q_{I1} \neq q_{I2}$) that satisfy the following constraints:*

- *$\mathcal{A}_1$ and $\mathcal{A}_2$ are self-loop alternating automata simplified in the sense of Corollary 2.3.20 (i.e., for all transitions $t \in \Delta_1 \cup \Delta_2$, the set of acceptance conditions of t is not empty only if t is a self-loop)*
- *$\mathcal{A}_1^{q,\mathcal{F}_1\cup\mathcal{F}_2} = \mathcal{A}_2^{q,\mathcal{F}_1\cup\mathcal{F}_2}$ holds for all states $q \in Q_1 \cap Q_2$;*
- *for all indices $n \in \{1, 2\}$, the automaton $\mathcal{A}_n$ is an acceptance closed alternating automaton, and if $\mathcal{A}_n$ has an f-state for some $f \in \mathcal{F}_n$, then $\mathcal{A}_n$ has an f-representative state; and*
- *if both $\mathcal{A}_1$ and $\mathcal{A}_2$ have an f-state for some $f \in \mathcal{F}_1 \cap \mathcal{F}_2$, then $\mathcal{A}_1$ and $\mathcal{A}_2$ share an f-representative state.*

Table 5.2: Definition of a new translation rule for the $\wedge$ connective (continuation of Table 3.1)

$\circ$	$\mathcal{F}_\circ$	$\Delta_\circ$
$\wedge$	$\emptyset$	$\left\{ \left\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q_1' \cup Q_2' \right\rangle \;\middle|\; \begin{array}{l} \langle q_{I1}, \theta_1, F_1, Q_1' \rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q_2' \rangle \in \Delta_2, \\ \{q_{I1}, q_{I2}\} \not\subseteq Q_1' \cup Q_2' \end{array} \right\} \cup \left\{ \begin{array}{l} \langle q_I, (\theta_1 \wedge \theta_2), F_1 \cup F_2, \\ (Q_1' \cup Q_2' \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\} \rangle \end{array} \;\middle|\; \begin{array}{l} \langle q_{I1}, \theta_1, F_1, Q_1' \rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q_2' \rangle \in \Delta_2, \\ \{q_{I1}, q_{I2}\} \subseteq Q_1' \cup Q_2' \end{array} \right\}$

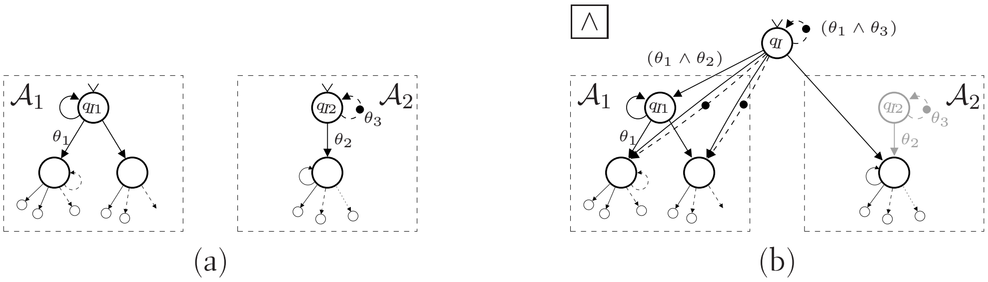


Fig. 5.5: New translation rule for the $\wedge$ connective. (a) Two component automata $\mathcal{A}_1$ and $\mathcal{A}_2$; (b) Automaton built from $\mathcal{A}_1$ and $\mathcal{A}_2$ using the refined rule for the $\wedge$ connective

Define the alternating automaton $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F} \rangle$*, where* $Q \overset{def}{=} Q_1 \cup Q_2 \cup \{q_I\}$ *(for a new state* $q_I \notin Q_1 \cup Q_2$*),*

$$\Delta \overset{def}{=} \Delta_1 \cup \Delta_2 \cup \left\{ \left\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q_1' \cup Q_2' \right\rangle \;\middle|\; \begin{array}{l} \langle q_{I1}, \theta_1, F_1, Q_1' \rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q_2' \rangle \in \Delta_2, \\ \{q_{I1}, q_{I2}\} \not\subseteq Q_1' \cup Q_2' \end{array} \right\}$$
$$\cup \left\{ \begin{array}{l} \langle q_I, (\theta_1 \wedge \theta_2), F_1 \cup F_2, \\ (Q_1' \cup Q_2' \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\} \rangle \end{array} \;\middle|\; \begin{array}{l} \langle q_{I1}, \theta_1, F_1, Q_1' \rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q_2' \rangle \in \Delta_2, \\ \{q_{I1}, q_{I2}\} \subseteq Q_1' \cup Q_2' \end{array} \right\}$$

and $\mathcal{F} \overset{def}{=} \mathcal{F}_1 \cup \mathcal{F}_2$*. The automaton* $\mathcal{A}$ *is a self-loop alternating automaton simplified in the sense of* Corollary 2.3.20*,* $\mathcal{A}^{q,\mathcal{F}_n} = \mathcal{A}_n^q$ *holds for all* $n \in \{1, 2\}$ *and* $q \in Q \cap Q_n$*, and for all* $w \in (2^{AP})^\omega$*,* $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ *holds iff* $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ *holds. Furthermore,* $\mathcal{A}$ *is an acceptance closed alternating automaton such that if* $\mathcal{A}$ *has an* f*-state for some* $f \in \mathcal{F}$*, then* $\mathcal{A}$ *has an* f*-representative state that is* f*-representative also in all automata* $\mathcal{A}_n$ *(*$n \in \{1, 2\}$*) that have an* f*-state.*

Observe that all automata $\mathcal{A}_1 = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1 \rangle$ and $\mathcal{A}_2 = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2 \rangle$ constructed from node subformulas of an LTL formula $\varphi \in LTL^{\text{PNF}}(AP)$ using the basic translation rules satisfy the assumptions given in Proposition 5.4.3. The automata obtained in this way are self-loop alternating automata such that $\mathcal{A}_1^{q,\mathcal{F}_1 \cup \mathcal{F}_2} = \mathcal{A}_2^{q,\mathcal{F}_1 \cup \mathcal{F}_2}$ holds for all $q \in Q_1 \cap Q_2$ by the discussion in Sect. 3.1.1, and for all $f \in \mathcal{F}_1 \cup \mathcal{F}_2$, the union of the state sets of $\mathcal{A}_1$ and $\mathcal{A}_2$ contains at most one f-state (which is the initial state of a subautomaton built for a strong temporal eventuality subformula of φ). Clearly, this f-state is trivially f-representative in any automaton to which it belongs.

We divide the proof of Proposition 5.4.3 into several lemmas. It is easy to see that $\mathcal{A}$ has the simple structural properties listed in the proposition.

Lemma 5.4.4 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}$ be three alternating automata as specified in Proposition 5.4.3. The automaton $\mathcal{A}$ is a self-loop alternating automaton (simplified in the sense of Corollary 2.3.20) such that $\mathcal{A}^{q,\mathcal{F}_n} = \mathcal{A}_n^q$ holds for all $n \in \{1,2\}$ and $q \in Q \cap Q_n$.*

Proof: Because $\mathcal{A}_1$ and $\mathcal{A}_2$ are self-loop alternating automata, it is easy to see that also the automaton $\mathcal{A}$ is a self-loop alternating automaton (all loops introduced in the definition of $\mathcal{A}$ are self-loops). Furthermore, because $\mathcal{A}_1$ and $\mathcal{A}_2$ are simplified in the sense of Corollary 2.3.20, then so is $\mathcal{A}$. If $q \in Q \cap Q_n$ holds for some $n \in \{1,2\}$, then $q \neq q_I$ holds, and because every transition in the subautomaton $\mathcal{A}^q$ is a transition of $\mathcal{A}_n$, it follows that $\mathcal{A}^{q,\mathcal{F}_n} = \mathcal{A}_n^q$ holds. □

The next result shows that every word fin-accepted by $\mathcal{A}$ is fin-accepted by both $\mathcal{A}_1$ and $\mathcal{A}_2$. (Actually, this language inclusion holds generally for all alternating automata $\mathcal{A}_1$ and $\mathcal{A}_2$.)

Lemma 5.4.5 *Let $\mathcal{A}_1 = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$, $\mathcal{A}_2 = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2\rangle$ and $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ be three alternating automata, where $q_I \notin Q_1 \cup Q_2$ holds, and Q (Δ, $\mathcal{F}$) is defined from Q_1 and Q_2 (Δ_1 and Δ_2, $\mathcal{F}_1$ and $\mathcal{F}_2$) as specified in Proposition 5.4.3. For all $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ holds.*

Proof: Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$, and let $G = \langle V, E, L\rangle$ be a fin-accepting run of $\mathcal{A}$ on w. By Proposition 2.3.7, there exists an index $0 \leq i \leq \omega$ and a chain of edges $(e_j)_{0 \leq j < i+1}$, $e_j \in E \cap (V_j \times 2^{V_{j+1}})$, such that the transition $L(e_j) \in \Delta$ is an initial self-loop of $\mathcal{A}$ for all $0 \leq j < i$, and if $i < \omega$, then the transition $L(e_i) \in \Delta$ is an initial transition of $\mathcal{A}$ that is not a self-loop.

Write $L(e_j) = t_j = \langle q_I, \theta_j, F_j, Q'_j\rangle$ for all $0 \leq j < i+1$, and fix $0 \leq j < i+1$. From the definition of $\mathcal{A}$ it follows that for all $k \in \{1,2\}$, there exists a transition $t_{j,k} = \langle q_{Ik}, \theta_{j,k}, F_{j,k}, Q'_{j,k}\rangle \in \Delta_k$ for some $\theta_{j,k} \in PL(AP)$, $F_{j,k} \subseteq \mathcal{F}_k$ and $Q'_{j,k} \subseteq Q_k$ such that $\theta_j = (\theta_{j,1} \wedge \theta_{j,2})$ holds. If $j < i$, then $L(e_j)$ is an initial self-loop of $\mathcal{A}$, in which case $F_j = F_{j,1} \cup F_{j,2}$ and $Q'_j = \big(Q'_{j,1} \cup Q'_{j,2} \cup \{q_I\}\big) \setminus \{q_{I1}, q_{I2}\}$ (where $\{q_{I1}, q_{I2}\} \subseteq Q'_{j,1} \cup Q'_{j,2}$). Otherwise, if $i < \omega$, then $F_i = \emptyset$ and $Q'_i = Q'_{i,1} \cup Q'_{i,2}$.

For all $0 \leq j < i$ and $k \in \{1,2\}$, write $Q'_{j,k} \setminus \{q_{I1}, q_{I2}\} = \{q_{j,k,1}, q_{j,k,2}, \ldots, q_{j,k,n_{j,k}}\}$ for some $0 \leq n_{j,k} < \omega$. If $i < \omega$, write $Q'_{i,k} = \{q_{i,k,1}, q_{i,k,2}, \ldots, q_{i,k,n_{i,k}}\}$ for some $0 \leq n_{i,k} < \omega$ ($k \in \{1,2\}$). We construct a fin-accepting semi-run $G' = \langle V', E', L'\rangle$ of $\mathcal{A}_1$ on w.

(Definition of G') Define the levels of G' inductively: let $V'_0 \overset{\text{def}}{=} \{v_{0,1}\}$ and $L'(v_{0,1}) \overset{\text{def}}{=} q_{I1}$, and assume that V'_j and $L'(v)$ have already been defined for some $0 \leq j < i+1$ and for all $v \in V'_j$, respectively. For all $v \in V'_j$, define a

set $S(v)$ of nodes (which will correspond to the successors of v in G') by

$$S(v) \stackrel{\text{def}}{=} \begin{cases} \emptyset & \text{if } L'(v) \notin \{q_{I1}, q_{I2}\} \\ \{v_{j,k,1}, \ldots, v_{j,k,n_{j,k}}\} & \text{if } L'(v) = q_{Ik}\ (k \in \{1,2\}), \\ \quad \cup \left\{v_{j+1,\ell} \,\middle|\, \ell \in \{1,2\}, q_{I\ell} \in Q'_{j,k}\right\} & \quad \text{and either } i = \omega, \text{ or } j < i \\ \{v_{j,k,1}, \ldots, v_{j,k,n_{j,k}}\} & \text{if } L'(v) = q_{Ik}\ (k \in \{1,2\}), \\ & \quad i < \omega, \text{ and } j = i. \end{cases}$$

Let $V'_{j+1} \stackrel{\text{def}}{=} \bigcup_{v \in V'_j} S(v)$. (We distinguish between nodes by their subscripts; thus, V'_{j+1} consists of at most $n_{j,1} + n_{j,2} + 2$ nodes not included in any previously defined level of G'.) Define the labeling for the nodes at level $j+1$ by setting

$$L'(v_{j,k,\ell}) \stackrel{\text{def}}{=} q_{j,k,\ell} \qquad \text{and} \qquad L'(v_{j+1,k}) \stackrel{\text{def}}{=} q_{Ik}$$

for all (relevant) combinations of $k \in \{1,2\}$ and $1 \le \ell \le n_{j,k}$. If $i < \omega$, let $V'_j \stackrel{\text{def}}{=} \emptyset$ for all $i+1 < j < \omega$. To complete the definition of G', let $E' \stackrel{\text{def}}{=} \left\{\left\langle v_{j,k}, S(v_{j,k})\right\rangle \,\middle|\, v_{j,k} \in V'_j \text{ for some } 0 \le j < i+1 \text{ and } k \in \{1,2\}\right\}$, and for all $e = \left\langle v_{j,k}, S(v_{j,k})\right\rangle \in E'$, let $L'(e) \stackrel{\text{def}}{=} t_{j,k}$.

(G' is a fin-accepting semi-run of $\mathcal{A}_1$ on w) We check that G' is a fin-accepting semi-run of $\mathcal{A}_1$ on w.

(Partitioning) Clearly, $V'_0 = \{v_{0,1}\}$ is a singleton, V' consists of finite disjoint levels, and all edges of E' lie between successive levels of G'.

(Causality) Let $v \in V'_j$ for some $0 \le j < \omega$. Then $v = v_{j-1,k,\ell}$ holds for some $k \in \{1,2\}$ and $1 \le \ell \le n_{j-1,k}$, or $v = v_{j,k}$ holds for some $k \in \{1,2\}$. In the former case, v has no outgoing edges; otherwise v has the unique outgoing edge $\left\langle v, S(v)\right\rangle \in E'$. It follows that G' satisfies the forward semi-causality constraint.

If $v' \in V'_j$ holds for some $1 \le j < \omega$, then $j < i+2$ holds, and, by the inductive definition of the levels of G', there exists a node $v \in V'_{j-1}$ such that $L'(v) \in \{q_{I1}, q_{I2}\}$ and $v' \in S(v)$ hold. Because $j - 1 < i + 1$ holds, $L'(v) \in \{q_{I1}, q_{I2}\}$ implies that $v = v_{j-1,k}$ for some $k \in \{1,2\}$. Because $\left\langle v_{j-1,k}, S(v_{j-1,k})\right\rangle \in E'$, it follows that v' is a successor of the node v at level $j-1$, and G' satisfies the backward causality constraint.

(Consistency of L') Clearly, $L'(V'_0) = \left\{L'(v_{0,1})\right\} = \{q_{I1}\}$ holds. Let $e = \left\langle v_{j,k}, S(v_{j,k})\right\rangle \in E'$ for some $0 \le j < i+1$ and $k \in \{1,2\}$. By the definition of G', e is labeled with the transition $t_{j,k} = \langle q_{Ik}, \theta_{j,k}, F_{j,k}, Q'_{j,k}\rangle \in \Delta_k$. Because the edge $e_j \in E \cap (V_j \times 2^{V_{j+1}})$ is labeled in G with the transition $t_j = \langle q_I, \theta_j, F_j, Q'_j\rangle \in \Delta$, $w(j) \models \theta_j = (\theta_{j,1} \wedge \theta_{j,2})$ holds (G is a run of $\mathcal{A}$). By the semantics of LTL, it follows that $w(j) \models \theta_{j,k}$ holds. Moreover, $L'(v_{j,k}) = q_{Ik}$ holds, and it is straightforward to check from the definitions that also $L'\big(S(v_{j,k})\big) = Q'_{j,k}$ holds. Thus the labeling L is consistent.

(Acceptance) If $V'_j = \emptyset$ holds for some $0 \le j < \omega$ (which is always the case if $i < \omega$ holds), then E' is finite. In this case $\mathcal{B}(G') = \emptyset$, and G' is trivially fin-accepting. Otherwise $i = \omega$, and G' contains an infinite branch $\beta = (e'_j)_{0 \le j < \omega} \in \mathcal{B}(G')$, where $e'_j \in E' \cap (V'_j \times 2^{V'_{j+1}})$ for all $0 \le j < \omega$. By the definition of E', $e'_j = \left\langle v_{j,k}, S(v_{j,k})\right\rangle$ holds for some

$k \in \{1,2\}$ for all $0 \leq j < \omega$, and thus $L'(e'_j) \in \{t_{j,1}, t_{j,2}\}$ holds for all $0 \leq j < \omega$. Suppose that $\mathrm{fin}(\beta) \neq \emptyset$ holds. Then there exists an acceptance condition $f \in \mathcal{F}$ and an index $0 \leq \ell < \omega$ such that the transition $L'(e'_j)$ is an f-transition for all $\ell \leq j < \omega$. Because $i = \omega$, each edge e_j in the run G is labeled with a self-loop of $\mathcal{A}$ for all $0 \leq j < \omega$, and thus the acceptance conditions of $L(e_j)$ include the acceptance conditions of the transitions $t_{j,1}$ and $t_{j,2}$ for all $0 \leq j < \omega$. But then $\mathrm{fin}\big((e_j)_{0 \leq j < \omega}\big) \neq \emptyset$ holds, contrary to the assumption that G is a fin-accepting run of $\mathcal{A}$. Therefore $\mathrm{fin}(\beta)$ is empty, and G' is a fin-accepting semi-run of $\mathcal{A}_1$ on w.

(G' can be extended to a fin-accepting run) Let $v \in V'$ be a node with no outgoing edges in G'. Then $v = v_{j,k,\ell} \in V'_{j+1}$ holds for some $0 \leq j < i+1$, $k \in \{1,2\}$ and $1 \leq \ell \leq n_{j,k}$, and $L'(v) \in Q'' \stackrel{\text{def}}{=} \{q_{j,k,1}, \ldots, q_{j,k,n_{j,k}}\} \subseteq Q_1$ (because G' is a run of $\mathcal{A}_1$, all nodes in V'_{j+1} are labeled with descendants of $L'(v_{0,1}) = q_{I1}$ by Proposition 2.3.6). In the run G, $e_j \in E \cap (V_j \times 2^{V_{j+1}})$ holds, and $L(e_j) = \langle q_I, \theta_j, F_j, Q'_j \rangle$, where $Q'' \subseteq Q'_j$. Because G is a fin-accepting run of $\mathcal{A}$ on w, $\mathcal{A}^q$ fin-accepts w^{j+1} for all $q \in Q'_j$, and because $Q'' \subseteq Q_1$ holds, $w^{j+1} \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^{q,\mathcal{F}_1}) = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1^q)$ holds for all $q \in Q''$ (Lemma 5.4.4). In particular, $\mathcal{A}_1^{L'(v)}$ fin-accepts w^{j+1}, and because this result holds for all $v \in V'$ with no outgoing edges, the semi-run G' can be extended into a fin-accepting run of $\mathcal{A}_1$ on w by Proposition 2.3.14. We conclude that $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)$ holds.

($w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$) By repeating the above construction of G' for the automaton $\mathcal{A}_2$ (i.e., by starting the inductive definition of the levels of V' from the set $V'_0 \stackrel{\text{def}}{=} \{v_{0,2}\}$ with $L'(v_{0,2}) \stackrel{\text{def}}{=} q_{I2}$), we obtain a fin-accepting run of $\mathcal{A}_2$ on w. Thus $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)$ and $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$ hold. □

To prove the converse language inclusion, we need the following technical result.

Lemma 5.4.6 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}$ be three alternating automata as specified in Proposition 5.4.3, and let $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$. There exists (by possibly switching the roles of $\mathcal{A}_1$ and $\mathcal{A}_2$) an index $0 \leq i \leq \omega$ and, for all $n \in \{1,2\}$, a sequence $(t_{j,n})_{0 \leq j < i+1} = \big(\langle q_{In}, \theta_{j,n}, F_{j,n}, Q'_{j,n} \rangle\big)_{0 \leq j < i+1} \in \Delta_n^{i+1}$ of initial transitions of $\mathcal{A}_n$ such that*

- *the sequence $(t_{j,1})_{0 \leq j < i+1}$ is a chain of initial transitions of $\mathcal{A}_1$ such that the transition $t_{j,1}$ is an initial self-loop of $\mathcal{A}_1$ for all $0 \leq j < i$;*
- *there exists a (possibly infinite) sequence of integers $0 = \hat{\imath}_0 < \hat{\imath}_1 < \cdots < i$ such that the sequence $(t_{j,2})_{0 \leq j < i+1}$ can be written either (i) as a finite concatenation $(t_{j,2})_{\hat{\imath}_0 \leq j < \hat{\imath}_1} (t_{j,2})_{\hat{\imath}_1 \leq j < \hat{\imath}_2} \cdots (t_{j,2})_{\hat{\imath}_\ell \leq j < i+1}$ ($0 \leq \ell < \omega$) of nonempty segments, the last one of which is infinite iff $i = \omega$ holds, or (ii) as an infinite concatenation $(t_{j,2})_{\hat{\imath}_0 \leq j < \hat{\imath}_1} (t_{j,2})_{\hat{\imath}_1 \leq j < \hat{\imath}_2} \cdots$ of nonempty finite segments, where (in both cases) every segment is a chain of initial transitions of $\mathcal{A}_2$, and every non-final segment ends in an initial non-self-loop of $\mathcal{A}_2$;*
- *$w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$ holds for all $0 \leq j < i+1$;*
- *for all $0 \leq j < i+1$, $n \in \{1,2\}$ and $q \in Q'_{j,n}$, $\mathcal{A}_n^q$ fin-accepts w^{j+1};*

- $\{q_{I1}, q_{I2}\} \subseteq Q'_{j,1} \cup Q'_{j,2}$ *holds for all* $0 \leq j < i$*;*
- *if* $i < \omega$ *holds, then* $\{q_{I1}, q_{I2}\} \not\subseteq Q'_{i,1} \cup Q'_{i,2}$ *holds; and*
- *if* $i = \omega$ *holds, then, for all* $f \in \mathcal{F}_1 \cup \mathcal{F}_2$*, there exist infinitely many indices* $0 \leq j < \omega$ *such that* $f \notin F_{j,1} \cup F_{j,2}$ *holds.*

Proof: Because $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ holds, $\mathcal{A}_1$ and $\mathcal{A}_2$ have fin-accepting runs $G^1 = \langle V^1, E^1, L^1 \rangle$ and $G^2 = \langle V^2, E^2, L^2 \rangle$ on w, respectively. (Without loss of generality, we may assume that $V^1 \cap V^2 = E^1 \cap E^2 = \emptyset$ holds.) Furthermore, because the automaton $\mathcal{A}_n$ ($n \in \{1, 2\}$) is an acceptance closed alternating automaton that has an f-representative state for all $f \in \mathcal{F}_n$ for which it has an f-state, we may assume that G^1 and G^2 are fin-acceptance synchronized (Proposition 4.3.5). As a matter of fact, because $\mathcal{A}_1$ and $\mathcal{A}_2$ share an f-representative state for all $f \in \mathcal{F}_1 \cap \mathcal{F}_2$ for which they both have an f-state, G^1 and G^2 can be built using the construction given in the proof of Proposition 4.3.5 such that there are, for all acceptance conditions $f \in \mathcal{F}_1 \cup \mathcal{F}_2$, infinitely many indices $0 \leq j < \omega$ such that no transition labeling an edge starting from a node at level j of G^1 or G^2 is an f-transition. Therefore, not only are the runs G^1 and G^2 individually fin-acceptance synchronized, but the graph obtained by juxtaposing G^1 and G^2 is similarly synchronized; we say that G^1 and G^2 are *mutually* fin-acceptance synchronized.

Let $n \in \{1, 2\}$. By Proposition 2.3.7, there exists an index $0 \leq i_n \leq \omega$ and a chain of edges $(e_{j,n})_{0 \leq j < i_n + 1}$ in G^n, $e_{j,n} \in E^n \cap (V^n_j \times 2^{V^n_{j+1}})$, such that for all $0 \leq j < i_n$, the transition $L^n(e_{j,n}) \in \Delta_n$ is an initial self-loop of $\mathcal{A}_n$, and if $i_n < \omega$ holds, then $L^n(e_{i_n,n})$ is an initial transition of $\mathcal{A}_n$ that is not a self-loop.

Without loss of generality, we may assume that $i_2 \leq i_1$ holds (if this is not the case, switch the roles of $\mathcal{A}_1$ and $\mathcal{A}_2$). For all $n \in \{1, 2\}$, let $t_{j,n} \stackrel{def}{=} L^n(e_{j,n}) = \langle q_{In}, \theta_{j,n}, F_{j,n}, Q'_{j,n} \rangle$ for all $0 \leq j < i_n + 1$. Clearly, for all $n \in \{1, 2\}$, the sequence $(t_{j,n})_{0 \leq j < i_n + 1}$ is a chain of initial transitions of $\mathcal{A}_n$. Because the edge $e_{j,n} \in E^n \cap (V^n_j \times 2^{V^n_{j+1}})$ is an edge in a fin-accepting run of $\mathcal{A}_n$ on w and L^n is consistent, $w(j) \models \theta_{j,n}$ holds for all $0 \leq j < i_n + 1$, $\mathcal{A}^q_n$ fin-accepts w^{j+1} for all $0 \leq j < i_n + 1$ and $q \in Q'_{j,n}$ (Proposition 2.3.9), and $q_{In} \in Q'_{j,n}$ holds for all $0 \leq j < i_n$.

If $i_2 = \omega$ holds, then the result obviously follows if we define $i \stackrel{def}{=} \omega$: in this case the sequence $(t_{j,2})_{0 \leq j < i+1}$ consists of a single infinite segment, and the fact that G^1 and G^2 are mutually fin-acceptance synchronized guarantees that $f \notin F_{j,1} \cup F_{j,2}$ holds for infinitely many $0 \leq j < \omega$ for all $f \in \mathcal{F}_1 \cup \mathcal{F}_2$.

If $i_2 < \omega$ holds, then clearly $(t_{j,2})_{0 \leq j < i_2+1}$ consists of a single segment conforming to the criteria given in the lemma, and this segment ends in an initial non-self-loop transition of $\mathcal{A}_2$. In this case we extend the sequence $(t_{j,2})_{0 \leq j < i_2+1}$ with another segment using the following inductive construction; we also define a sequence of integers $\hat{\imath}_1, \hat{\imath}_2, \ldots$. Let $\hat{\imath}_1 \stackrel{def}{=} i_2 + 1$.

Assume that $\hat{\imath}_k \leq i_1 + 1$ ($\hat{\imath}_k < \omega$) and $t_{j,2} = \langle q_{I2}, \theta_{j,2}, F_{j,2}, Q'_{j,2} \rangle$ have already been defined for some $1 \leq k < \omega$ and for all $0 \leq j < \hat{\imath}_k$, and the sequence $(t_{j,2})_{0 \leq j < \hat{\imath}_k}$ can be expressed as a finite concatenation of finite nonempty segments corresponding to chains of initial transitions of $\mathcal{A}_2$ ending in an initial transition that is not a self-loop, $w(j) \models \theta_{j,2}$ holds for all

$0 \leq j < \hat{\imath}_k$, and $\mathcal{A}_2^q$ fin-accepts w^{j+1} for all $0 \leq j < \hat{\imath}_k$ and $q \in Q'_{j,2}$. (By the above definition of $(t_{j,2})_{0 \leq j < i_2+1}$, the assumption clearly holds for $k = 1$.)

If $\{q_{I1}, q_{I2}\} \not\subseteq Q'_{\hat{\imath}_k-1,1} \cup Q'_{\hat{\imath}_k-1,2}$ holds, we abort the inductive construction: in this case, the result follows by defining $i \stackrel{def}{=} \hat{\imath}_k - 1$.

Otherwise, if $\{q_{I1}, q_{I2}\} \subseteq Q'_{\hat{\imath}_k-1,1} \cup Q'_{\hat{\imath}_k-1,2}$ holds, then, because $t_{\hat{\imath}_k-1,2}$ is not an initial self-loop of $\mathcal{A}_2$, $q_{I2} \notin Q'_{\hat{\imath}_k-1,2}$ holds, and thus $q_{I2} \in Q'_{\hat{\imath}_k-1,1}$ holds. Suppose that $q_{I1} \notin Q'_{\hat{\imath}_k-1,1}$ holds (in which case $\hat{\imath}_k - 1 = i_1 < \omega$); then obviously $q_{I1} \in Q'_{\hat{\imath}_k-1,2}$ holds. It follows that $\{q_{I1}, q_{I2}\} \subseteq Q_1 \cap Q_2$ holds, and because $q_{I1} \neq q_{I2}$ and $\mathcal{A}_1^{q_{I1},\mathcal{F}_1\cup\mathcal{F}_2} = \mathcal{A}_2^{q_{I1},\mathcal{F}_1\cup\mathcal{F}_2}$ hold by the assumptions given in Proposition 5.4.3, the automaton $\mathcal{A}_1$ contains the simple cycle (q_{I1}, q_{I2}, q_{I1}) that is not a self-loop. The existence of this cycle contradicts the assumption that $\mathcal{A}_1$ is a self-loop alternating automaton. Therefore it must be the case that $q_{I1} \in Q'_{\hat{\imath}_k-1,1}$ holds, and thus $\{q_{I1}, q_{I2}\} \subseteq Q'_{\hat{\imath}_k-1,1}$.

Because G^1 is a fin-accepting run of $\mathcal{A}_1$ and $q_{I2} \in Q'_{\hat{\imath}_k-1,1}$ holds, it follows that we can extract from G^1 a fin-accepting run $(G^2)' = \langle (V^2)', (E^2)', (L^2)' \rangle$ of $\mathcal{A}_2^{q_{I2}}$ (which is a fin-accepting run of $\mathcal{A}_2$ by Proposition 2.3.12) on $w^{\hat{\imath}_k}$. By Proposition 2.3.7, there now exists an integer $0 \leq i'_2 < \omega$ such that the run $(G^2)'$ contains a chain of edges $(e_j)_{0 \leq j < i'_2+1}$, $e_j \in (E^2)' \cap \big((V^2)'_j \times 2^{(V^2)'_{j+1}}\big)$, labeled with initial transitions of $\mathcal{A}_2$ such that the transition $(L^2)'(e_j) = \langle q_{I2}, \theta_j, F_j, Q'_j \rangle$ is an initial self-loop of $\mathcal{A}_2$ for all $0 \leq j < i'_2$, and if $i'_2 < \omega$ holds, then $(L^2)'(e_{i'_2}) = \langle q_{I2}, \theta_{i'_2}, F_{i'_2}, Q'_{i'_2} \rangle$ is not a self-loop of $\mathcal{A}_2$. Furthermore, because $(G^2)'$ is embedded in the fin-accepting run G^1, $e_{j-\hat{\imath}_k} \in E^1 \cap (V^1_j \times 2^{V^1_{j+1}})$ and $w(j) \models \theta_{j-\hat{\imath}_k}$ hold for all $\hat{\imath}_k \leq j < \hat{\imath}_k + i'_2 + 1$, $w^{j+1} \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1^q) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q,\mathcal{F}_2}) = \mathcal{L}_{\text{fin}}(\mathcal{A}_2^q)$ holds for all $\hat{\imath}_k \leq j < \hat{\imath}_k + i'_2 + 1$ and $q \in Q'_{j-\hat{\imath}_k}$, and $q_{I2} \in Q'_{j-\hat{\imath}_k}$ holds for all $\hat{\imath}_k \leq j < \hat{\imath}_k + i'_2$.

If $\hat{\imath}_k + i'_2 \leq i_1$ holds, then, if $i'_2 = \omega$ holds, then also $i_1 = \omega$ holds. In this case the result follows by defining $i \stackrel{def}{=} \omega$ and $t_{j,2} \stackrel{def}{=} (L^2)'(e_{j-\hat{\imath}_k})$ for all $\hat{\imath}_k \leq j < \omega$ (because $(G^2)'$ is embedded in the fin-acceptance synchronized run G^1, it is easy to see that there again exist infinitely many $0 \leq j < \omega$ such that neither $t_{j,1}$ nor $t_{j,2}$ is an f-transition). If $i'_2 < \omega$ holds, define $\hat{\imath}_{k+1} \stackrel{def}{=} \hat{\imath}_k + i'_2 + 1$, and $t_{j,2} \stackrel{def}{=} (L^2)'(e_{j-\hat{\imath}_k})$ for all $\hat{\imath}_k \leq j < \hat{\imath}_{k+1}$. Clearly, the assumptions of the inductive construction now hold for $\hat{\imath}_{k+1}$. In this case we continue the construction.

Otherwise, if $\hat{\imath}_k + i'_2 > i_1$ holds, then $i_1 < \omega$ holds. We claim that the result now follows by defining $i \stackrel{def}{=} i_1$ and $t_{j,2} \stackrel{def}{=} (L^2)'(e_{j-\hat{\imath}_k})$ for all $\hat{\imath}_k \leq j \leq i_1$. It is easy to see that all except the second to last condition in the lemma hold directly by construction of the sequences $(t_{j,1})_{0 \leq j < i+1}$ and $(t_{j,2})_{0 \leq j < i+1}$. Assume that $\{q_{I1}, q_{I2}\} \subseteq Q'_{i,1} \cup Q'_{i,2}$ holds. Because the transition $t_{i,1}$ is not an initial self-loop of $\mathcal{A}_1$, it follows that $q_{I1} \in Q'_{i,2}$ holds. But then it again follows that the automaton $\mathcal{A}_1$ contains the cycle (q_{I1}, q_{I2}, q_{I1}), a contradiction. We conclude that $\{q_{I1}, q_{I2}\} \not\subseteq Q'_{i,1} \cup Q'_{i,2}$ holds.

In summary, if the above inductive construction ends after a finite number of steps, then the transition sequence $(t_{j,2})_{0 \leq j < i+1}$ consists of a finite number of segments (the last one of which may be infinite), and the sequences $(t_{j,1})_{0 \leq j < i+1}$ and $(t_{j,2})_{0 \leq j < i+1}$ satisfy the constraints given in the lemma by the above discussion. Otherwise the sequence $(t_{j,2})_{0 \leq i < i+1}$ is formed as the concatenation of infinitely many finite segments, all except the first one are embedded in the run G^1. Because G^1 is fin-acceptance synchronized, it is easy to see that for all $f \in \mathcal{F}_1 \cup \mathcal{F}_2$, there are again infinitely many indices $0 \leq j < \omega$ such that neither $t_{j,1}$ or $t_{j,2}$ is an f-transition, and thus all constraints of the lemma are satisfied also in this case. $\square$

We can now show that, for any automata $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}$ specified in Proposition 5.4.3, every word fin-accepted by both $\mathcal{A}_1$ and $\mathcal{A}_2$ is also accepted by the automaton $\mathcal{A}$.

Lemma 5.4.7 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}$ be three alternating automata as specified in Proposition 5.4.3. For all $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds.*

Proof: Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$. By Lemma 5.4.6, there exists an index $0 \leq i \leq \omega$ and two sequences of transitions $(t_{j,1})_{0 \leq j < i+1} \in \Delta_1^{i+1}$ and $(t_{j,2})_{0 \leq j < i+1} \in \Delta_2^{i+1}$ such that the conditions in the lemma are satisfied; for all $k \in \{1, 2\}$, write $t_{j,k} = \langle q_{Ik}, \theta_{j,k}, F_{j,k}, Q'_{j,k} \rangle$. We construct a fin-accepting semi-run $G = \langle V, E, L \rangle$ of $\mathcal{A}$ on w.

(Definition of G) For all $0 \leq j < i$ and $k \in \{1, 2\}$, write $Q'_{j,k} \setminus \{q_{I1}, q_{I2}\} = \{q_{j,k,1}, \ldots, q_{j,k,n_{j,k}}\}$ for some $0 \leq n_{j,k} < \omega$, and if $i < \omega$ holds, write $Q'_{i,k} = \{q_{i,k,1}, \ldots, q_{i,k,n_{i,k}}\}$ for some $0 \leq n_{i,k} < \omega$. Define the components of G by setting

- $V_0 \stackrel{\text{def}}{=} \{v_0\}$,
  $V_{j+1} \stackrel{\text{def}}{=} \{v_{j,1,1}, \ldots, v_{j,1,n_{j,1}}\} \cup \{v_{j,2,1}, \ldots, v_{j,2,n_{j,2}}\} \cup \begin{cases} \{v_{j+1}\} & \text{if } j < i \\ \emptyset & \text{otherwise} \end{cases}$
  for all $0 \leq j < i + 1$, and if $i < \omega$, let $V_j \stackrel{\text{def}}{=} \emptyset$ for all $i + 1 < j < \omega$;
- $E \stackrel{\text{def}}{=} \bigcup_{0 \leq j < i+1} \{\langle v_j, V_{j+1} \rangle\}$; and
- for all $0 \leq j < i + 1$, let $L(v_j) \stackrel{\text{def}}{=} q_I$ and $L(v_{j,k,\ell}) \stackrel{\text{def}}{=} q_{j,k,\ell}$ for all $k \in \{1, 2\}$ and $1 \leq \ell \leq n_{j,k}$. Furthermore, let $L(\langle v_j, V_{j+1} \rangle) \stackrel{\text{def}}{=} \langle q_I, (\theta_{j,1} \wedge \theta_{j,2}), F_{j,1} \cup F_{j,2}, (Q'_{j,1} \cup Q'_{j,2} \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\} \rangle$ for all $0 \leq j < i$, and if $i < \omega$ holds, let $L(\langle v_i, V_{i+1} \rangle) \stackrel{\text{def}}{=} \langle q_I, (\theta_{i,1} \wedge \theta_{i,2}), \emptyset, Q'_{i,1} \cup Q'_{i,2} \rangle$.

(G is a fin-accepting semi-run of $\mathcal{A}$ on w) We check that G is a fin-accepting semi-run of $\mathcal{A}$ on w.

(Partitioning) By the definition of G, $V_0 = \{v_0\}$ is a singleton, V consists of finite disjoint levels, and E is a collection of edges between consecutive levels of G.

(Causality) Let $v \in V_j$ for some $0 \leq j < \omega$. Then v either has no outgoing edges, or $v = v_j$ has the unique outgoing edge $\langle v_j, V_{j+1} \rangle \in E$. On the other hand, every node $v \in V_j$ for some $1 \leq j < \omega$ is a successor of the node $v_{j-1} \in V$. It follows that G satisfies the forward semi-causality and backward causality constraints.

(Consistency of L) Clearly, $L(v_0) = q_I$ holds. Let $0 \leq j < i + 1$, and let $\langle v_j, V_{j+1} \rangle \in E$. Because the transition sequences $(t_{j,1})_{0 \leq j < i+1}$ and $(t_{j,2})_{0 \leq j < i+1}$ satisfy the conditions given in Lemma 5.4.6, $w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$ holds. Because $t_{j,k}$ is an initial transition of $\mathcal{A}_k$ for all $k \in \{1, 2\}$, it follows from the definition of $\mathcal{A}$ that Δ contains a transition

$t = \langle q_I, (\theta_{j,1} \wedge \theta_{j,2}), F, Q' \rangle$, where either $F = F_{j,1} \cup F_{j,2}$ and $Q' = (Q'_{j,1} \cup Q'_{j,2} \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\}$ hold, or $F = \emptyset$ and $Q' = Q'_{j,1} \cup Q'_{j,2}$ (by the conditions given in Lemma 5.4.6, the latter case occurs iff $i < \omega$ and $j = i$ hold). From the definition of L it now follows that $L\big(\langle v_j, V_{j+1} \rangle\big) = t$, $L(v_j) = q_I$ and $L(V_{j+1}) = Q'$ hold, and thus the labeling L is consistent.

(Acceptance) If $i < \omega$ holds, then $\mathcal{B}(G) = \emptyset$ holds, and thus G is trivially fin-accepting. Otherwise G contains the unique infinite branch $\beta = \big(\langle v_j, V_{j+1} \rangle\big)_{0 \le j < \omega}$. By the definition of L, the transition $L\big(\langle v_j, V_{j+1} \rangle\big)$ has the set $F_{j,1} \cup F_{j,2}$ as its acceptance conditions for all $0 \le j < \omega$. Because there are, for all acceptance conditions $f \in \mathcal{F}_1 \cup \mathcal{F}_2$, infinitely many $0 \le j < \omega$ such that $f \notin F_{j,1} \cup F_{j,2}$ holds (Lemma 5.4.6), it is easy to see that $\mathrm{fin}(\beta) = \emptyset$ holds, and thus G is fin-accepting.

We conclude that G is a fin-accepting semi-run of $\mathcal{A}$ on w.

(G can be extended into a fin-accepting run) If $v \in V$ is a node with no outgoing edges in G, then $v = v_{j,k,\ell}$ holds for some $0 \le j < i+1$, $k \in \{1, 2\}$ and $1 \le \ell \le n_{j,k}$, and $L(v) = q_{j,k,\ell} \in Q'_{j,k}$ holds. By the assumptions given in Lemma 5.4.6, $w^{j+1} \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_k^q) = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_k^{q,\mathcal{F}_1 \cup \mathcal{F}_2}) = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$ holds for all $q \in Q'_{j,k}$. Thus, in particular, $w^{j+1} \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^{L(v)})$ holds. Because the same result holds for all nodes of G with no outgoing edges, the semi-run G can be extended into a fin-accepting run of $\mathcal{A}$ on w by Proposition 2.3.14, and thus $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A})$ holds. □

Lemma 5.4.5 and Lemma 5.4.7 show that the automaton $\mathcal{A}$ defined from two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ as specified in Proposition 5.4.3 fin-recognizes exactly the set intersection of the languages recognized by $\mathcal{A}_1$ and $\mathcal{A}_2$. Our last result in this section completes the proof of Proposition 5.4.3 by showing that the rule defined in the proposition preserves the existence of representative states for acceptance conditions.

Lemma 5.4.8 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}$ be three alternating automata as specified in Proposition 5.4.3. The automaton $\mathcal{A}$ is an acceptance closed alternating automaton such that if $\mathcal{A}$ has an f-state for some $f \in \mathcal{F}$, then $\mathcal{A}$ has an f-representative state that is f-representative also in all automata $\mathcal{A}_n$ ($n \in \{1, 2\}$) that have an f-state.*

Proof: ($\mathcal{A}$ is acceptance closed) Let $f \in \mathcal{F}$ be an acceptance condition, and let $t = \langle q, \theta, F, Q' \rangle \in \Delta$ be a transition of $\mathcal{A}$ such that $f \in F$ holds. Clearly, the state $q \in Q$ is an f-state of $\mathcal{A}$. Because $\mathcal{A}$ is simplified in the sense of Corollary 2.3.20 (Lemma 5.4.4), $F \neq \emptyset$ implies that t is a self-loop of $\mathcal{A}$, i.e., $q \in Q'$ holds, and thus t is f-closed. Because the same reasoning applies to all acceptance conditions and transitions of $\mathcal{A}$, it follows that $\mathcal{A}$ is an acceptance closed alternating automaton.

($\mathcal{A}$ has representative states for acceptance conditions) Let $f \in \mathcal{F}$ be an acceptance condition, and let $q \in Q = Q_1 \cup Q_2 \cup \{q_I\}$ be an f-state of $\mathcal{A}$. Then there exists an index $n \in \{1, 2\}$ such that either $q \in Q_n$ holds, or $q = q_I$, and the initial state of $\mathcal{A}_n$ is an f-state (if no initial transition of $\mathcal{A}_1$ or $\mathcal{A}_2$ were an f-transition, then no initial transition of $\mathcal{A}$ would be

such a transition, either). In either case, the automaton $\mathcal{A}_n$ contains an f-state, and it follows from the assumptions that the automaton $\mathcal{A}_n$ has an f-representative state $q_f \in Q_n$. Furthermore, if also $\mathcal{A}_{3-n}$ has an f-state, we may assume that $q_f \in Q_1 \cap Q_2$ is f-representative in $\mathcal{A}_1$ and $\mathcal{A}_2$. We show that q_f is f-representative in $\mathcal{A}$; that is, for all f-states $q \in Q$ and infinite words $w \in (2^{AP})^\omega$,

- if $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds, then $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$ holds, and
- if $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$ holds, then $w^i \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f})$ holds for some $0 \leq i < \omega$.

Let $q \in Q$ be an f-state of $\mathcal{A}$. If $q \neq q_I$ holds, then $q \in Q_n$ holds for some $n \in \{1, 2\}$. Because the state $q_f \in Q_n$ is an f-representative state of $\mathcal{A}_n$ by assumption, the above two conditions hold in $\mathcal{A}_n$ for all $w \in (2^{AP})^\omega$ with respect to the state q. Because $\mathcal{A}^{q'}$ fin-accepts w (by avoiding an initial f-transition) iff $\mathcal{A}^{q',\mathcal{F}_n}$ ($= \mathcal{A}^{q'}_n$, Lemma 5.4.4) fin-accepts w (by avoiding an initial f-transition) for all $q' \in Q_n$, it is easy to see that the conditions hold also in $\mathcal{A}$ with respect to the state q.

If $q = q_I$ and $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ hold, then, because obviously $\mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I}) = \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds (Proposition 2.3.12), $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q_{I1}})$ and $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_2) = \mathcal{L}_{\text{fin}}(\mathcal{A}_2^{q_{I2}})$ hold by Lemma 5.4.5 (and Proposition 2.3.12). By Proposition 2.3.15, there exist transitions $t_1 = \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1$ and $t_2 = \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2$ such that for all $n \in \{1, 2\}$, $w(0) \models \theta_n$ holds, and $\mathcal{A}^{q'}_n$ fin-accepts w^1 for all $q' \in Q'_n$.

Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$, and let $t_1 \in \Delta_1$ and $t_2 \in \Delta_2$ be initial transitions of $\mathcal{A}_1$ and $\mathcal{A}_2$ (respectively) having the above properties. From the definition of $\mathcal{A}$ it follows that $\mathcal{A}$ contains the transition $t = \big\langle q_I, (\theta_1 \wedge \theta_2), F, Q'\big\rangle$, where either (i) $F = \emptyset$ and $Q' = Q'_1 \cup Q'_2$ ($\{q_{I1}, q_{I2}\} \not\subseteq Q'_1 \cup Q'_2$), or (ii) $F = F_1 \cup F_2$ and $Q' = \big(Q'_1 \cup Q'_2 \cup \{q_I\}\big) \setminus \{q_{I1}, q_{I2}\}$ hold ($\{q_{I1}, q_{I2}\} \subseteq Q'_1 \cup Q'_2$). Clearly, because $w(0) \models \theta_n$ holds for all $n \in \{1, 2\}$, it is easy to see that $w(0) \models (\theta_1 \wedge \theta_2)$ holds. We show that the state q_f satisfies the criteria required of an f-representative state of $\mathcal{A}$ with respect to the state q_I.

<u>($w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ *implies* $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_I})$)</u> Suppose that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ holds. If the transition t is an f-transition of $\mathcal{A}$, then the transition t_n is an f-transition for some $n \in \{1, 2\}$, and thus the initial state $q_{In} \in Q_n$ is an f-state of $\mathcal{A}_n$. Therefore $q_f \in Q_n$ holds, and it follows by the above discussion that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}_n) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_{In}}_n)$ holds. Because q_f is an f-representative state of $\mathcal{A}_n$, $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_{In}}_n)$ holds, and thus we may change the transition t_n to another initial transition of $\mathcal{A}_n$ that does not include the condition f in its acceptance conditions (by applying Proposition 2.3.15).

We may thus assume that the transition $t = \big\langle q_I, (\theta_1 \wedge \theta_2), F, Q'\big\rangle$ defined above is not an f-transition of $\mathcal{A}$.

Clearly, because $q' \in Q'_1 \cup Q'_2$ holds for all $q' \in Q' \setminus \{q_I\}$, it follows (by the choice of the transitions t_1 and t_2) that the subautomaton $\mathcal{A}^{q'}$ (which coincides with $\mathcal{A}_1^{q',\mathcal{F}_1\cup\mathcal{F}_2}$ or $\mathcal{A}_2^{q',\mathcal{F}_1\cup\mathcal{F}_2}$) fin-accepts w^1 for all $q' \in Q' \setminus \{q_I\}$. If $q_I \in Q'$ holds, then $\{q_{I1}, q_{I2}\} \subseteq Q'_1 \cup Q'_2$ holds, and thus $w^1 \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_{I1}}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_{I2}})$ holds. It now follows by Proposition 2.3.12 and Lemma 5.4.7 that $w^1 \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ holds. Therefore, $\mathcal{A}^{q'}$ fin-accepts

w^1 for all $q' \in Q'$, and because $w(0) \models (\theta_1 \wedge \theta_2)$ and $f \notin F$ hold, it follows by Proposition 2.3.15 that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_I})$ holds.

$\underline{(w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_I}) \textit{ implies } w^i \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}) \textit{ for some } 0 \leq i < \omega)}$ Suppose that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_I})$ holds. Because q_I is an f-state of $\mathcal{A}$, q_{In} is an f-state of $\mathcal{A}_n$ for some $n \in \{1, 2\}$. Because $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_n^{q_{In}})$ holds, $\mathcal{A}_n^{q_{In}}$ has a fin-accepting run $G = \langle V, E, L\rangle$ on w. By Proposition 2.3.7, there exists an index $0 \leq i \leq \omega$ such that this run contains a chain of edges $(e_j)_{0 \leq j < i+1}$, $e_j \in E \cap (V_j \times 2^{V_{j+1}})$, labeled with initial transitions of $\mathcal{A}_n$, and if $i < \omega$ holds, then the transition $L(e_i)$ is not a self-loop. We claim that $w^k \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}_n^{q_{In}})$ holds for some $0 \leq k < \omega$. This is clear if $i < \omega$ holds, because in this case the non-self-loop transition $L(e_i)$ has an empty set of acceptance conditions ($\mathcal{A}_n$ is simplified in the sense of Corollary 2.3.20), and we may choose $k \stackrel{\text{def}}{=} i$ in this case. Otherwise $(e_j)_{0 \leq j < i+1}$ is an infinite branch in G labeled with initial self-loops of $\mathcal{A}_n$. Because G is fin-accepting, it follows that there necessarily exists an index $0 \leq k < \omega$ such that $L(e_k)$ is not an f-transition of $\mathcal{A}_n$, and $w^k \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}_n^{q_{In}})$ holds.

Because q_{In} is an f-state of $\mathcal{A}_n$, $q_f \in Q_n$ holds, and because q_f is f-representative, $w^k \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}_n^{q_{In}})$ implies that $(w^k)^\ell \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}_n^{q_f})$ holds for some $0 \leq \ell < \omega$. Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_n^{q_f}) = \mathcal{L}_{\text{fin}}(\mathcal{A}_n^{q_f, \mathcal{F}_1 \cup \mathcal{F}_2}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_f})$ holds, it follows that $w^{k+\ell} \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f})$ holds.

We conclude that q_f is an f-representative state of $\mathcal{A}$. By the choice of q_f, q_f is f-representative also in $\mathcal{A}_n$ ($n \in \{1, 2\}$) if $\mathcal{A}_n$ contains an f-state. Because the same holds for all acceptance conditions $f \in \mathcal{F}$, the result follows. $\square$

Proposition 5.4.3 now follows by combining Lemma 5.4.4, Lemma 5.4.5, Lemma 5.4.7 and Lemma 5.4.8. Due to the properties of the automata built using the basic translation rules and Proposition 5.4.3, it follows that the translation procedure presented in Sect. 3.1 remains correct in the sense of Theorem 3.3.2 if the set of basic translation rules is extended with the rule presented in Proposition 5.4.3. Furthermore, because this rule is closed under the construction of acceptance closed automata with representative states for their acceptance conditions, the automaton obtained from an LTL formula via the refined translation procedure can be translated into a nondeterministic automaton without introducing new acceptance conditions in the translation (Corollary 4.3.7). Further effects of using the new translation rule for the $\wedge$ connective will be discussed in Sect. 5.6.

5.5 BINARY TEMPORAL CONNECTIVES

In this section we refine the basic translation rules for the Until connectives and use them to derive new rules also for the Release connectives. Unlike the universally applicable translation rule proposed in the previous section for the $\wedge$ connective, however, the rules for the Until connectives apply only under a certain language containment relationship between the top-level subformulas of an Until formula. The optimized rules for the Until connectives nevertheless lead to improved translation rules for the Release connectives that do not necessitate checking for any language containment relationships between top-level subformulas (certain relationships admit further optimizations, however).

The Until Connectives

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two alternating automata built for the LTL formulas $\varphi_1 \in LTL^{\text{PNF}}(AP)$ and $\varphi_2 \in LTL^{\text{PNF}}(AP)$, respectively. To build an automaton for a formula of the form $(\varphi_1 \mathsf{U} \varphi_2)$, the basic translation rules for the U connectives take a new state (the initial state of the automaton to be built) and transform the initial transitions of $\mathcal{A}_1$ into self-loops starting from this state. Clearly, all initial self-loops of $\mathcal{A}_1$ are unrolled into initial transitions of the compound automaton in the application of the translation rule, and thus the initial state q_{I1} of $\mathcal{A}_1$ will remain reachable from the initial state of any automaton built from the compound automaton in subsequent translation steps. However, if the languages of φ_1 and φ_2 satisfy the language containment relationship $\mathcal{L}(\varphi_2) \subseteq \mathcal{L}(\varphi_1)$, it proves to be possible to avoid the introduction of self-loops having the initial state of $\mathcal{A}_1$ in their target states in the compound automaton by a slight modification to the way in which the initial self-loops of this automaton are defined. Similarly to the new translation rule defined for the $\wedge$ connective in the previous section, it may be possible to drop the state q_{I1} from the final automaton obtained at the end of the translation.

The new translation rules for the U connectives are based on the following result.

Proposition 5.5.1 *Let $\mathcal{A}_1 = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$ and $\mathcal{A}_2 = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2\rangle$ be two alternating automata satisfying the following constraints:*

- *$\mathcal{A}_1$ and $\mathcal{A}_2$ are self-loop alternating automata simplified in the sense of Corollary 2.3.20 (i.e., for all transitions $t \in \Delta_1 \cup \Delta_2$, the set of acceptance conditions of t is not empty only if t is a self-loop)*
- *$\mathcal{A}_1^{q,\mathcal{F}_1\cup\mathcal{F}_2} = \mathcal{A}_2^{q,\mathcal{F}_2\cup\mathcal{F}_2}$ holds for all states $q \in Q_1 \cap Q_2$;*
- *for all indices $n \in \{1, 2\}$, the automaton $\mathcal{A}_n$ is an acceptance closed alternating automaton, and if $\mathcal{A}_n$ has an f-state for some $f \in \mathcal{F}_n$, then $\mathcal{A}_n$ has an f-representative state;*
- *if both $\mathcal{A}_1$ and $\mathcal{A}_2$ have an f-state for some $f \in \mathcal{F}_1 \cap \mathcal{F}_2$, then $\mathcal{A}_1$ and $\mathcal{A}_2$ share an f-representative state; and*
- *$\mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds.*

Define the alternating automaton $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ obtained from $\mathcal{A}_1$ and $\mathcal{A}_2$ by setting $Q \stackrel{def}{=} Q_1 \cup Q_2 \cup \{q_I\}$ (where q_I is a new state not included in $Q_1 \cup Q_2$), $\Delta \stackrel{def}{=} \Delta_1 \cup \Delta_2 \cup \Delta'$, and $\mathcal{F} \stackrel{def}{=} \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}'$, where either

$$\begin{aligned} &\mathcal{F}' \stackrel{def}{=} \{f\} \text{ for a new acceptance condition } f \notin \mathcal{F}_1 \cup \mathcal{F}_2\text{, and} \\ &\Delta' \stackrel{def}{=} \big\{\langle q_I, \theta, \{f\}, (Q' \setminus \{q_{I1}\}) \cup \{q_I\}\rangle \,\big|\, \langle q_{I1}, \theta, F, Q'\rangle \in \Delta_1\big\} \\ &\qquad \cup \big\{\langle q_I, \theta, \emptyset, Q'\rangle \,\big|\, \langle q_{I2}, \theta, F, Q'\rangle \in \Delta_2\big\} \end{aligned}$$

or

$$\begin{aligned}\mathcal{F}' &\stackrel{def}{=} \emptyset, \text{ and}\\ \Delta' &\stackrel{def}{=} \big\{\langle q_I, \theta, F, (Q' \setminus \{q_{I1}\}) \cup \{q_I\}\rangle \,\big|\, \langle q_{I1}, \theta, F, Q'\rangle \in \Delta_1\big\}\\ &\quad \cup \big\{\langle q_I, \theta, \emptyset, Q'\rangle \,\big|\, \langle q_{I2}, \theta, F, Q'\rangle \in \Delta_2\big\}\end{aligned}$$

The automaton $\mathcal{A}$ is an acceptance closed self-loop alternating automaton that is simplified in the sense of Corollary 2.3.20*, $\mathcal{A}^{q,\mathcal{F}_n} = \mathcal{A}_n^q$ holds for all $n \in \{1,2\}$ and $q \in Q \cap Q_n$, and for all $w \in (2^{AP})^\omega$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds iff (i) there exists an index $0 \le i < \omega$ such that $w^i \in \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ holds and $w^j \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds for all $0 \le j < i$, or (ii) $\mathcal{F}' = \emptyset$, and $w^i \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds for all $0 \le i < \omega$. Furthermore, if $\mathcal{A}$ has an f-state for some $f \in \mathcal{F}$, then $\mathcal{A}$ has an f-representative state that is f-representative also in all automata $\mathcal{A}_n$ ($n \in \{1,2\}$) that have an f-state.*

Similarly to the previous section, we prove Proposition 5.5.1 in several steps. We first show that the automaton $\mathcal{A}$ has the simple structural properties listed in the proposition.

Lemma 5.5.2 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}$ be three alternating automata as specified in* Proposition 5.5.1*. The automaton $\mathcal{A}$ is a self-loop alternating automaton simplified in the sense of* Corollary 2.3.20 *such that $\mathcal{A}^{q,\mathcal{F}_n} = \mathcal{A}_n^q$ holds for all $n \in \{1,2\}$ and $q \in Q \cap Q_n$.*

Proof: Because $\mathcal{A}_1$ and $\mathcal{A}_2$ are self-loop alternating automata simplified in the sense of Corollary 2.3.20, it is easy to see that also $\mathcal{A}$ is the same kind of self-loop alternating automaton (all loops introduced into $\mathcal{A}$ by the definition are self-loops, and all non-self-loop transitions have an empty set of acceptance conditions). Let $q \in Q \cap Q_n$ for some $n \in \{1,2\}$. Then $q \neq q_I$ holds, and all transitions of the subautomaton $\mathcal{A}^q$ are transitions of $\mathcal{A}_n$. Therefore, the acceptance conditions of these transitions are subsets of $\mathcal{F}_n$, and it follows that $\mathcal{A}^{q,\mathcal{F}_n} = \mathcal{A}_n^q$ holds. □

The following lemma shows that the automaton $\mathcal{A}$ fin-recognizes the desired language. (This result is analogous to Lemma 3.3.1.)

Lemma 5.5.3 *Let $\mathcal{A}_1 = \langle 2^{AP}, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1\rangle$, $\mathcal{A}_2 = \langle 2^{AP}, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2\rangle$ and $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$ be three alternating automata such that $\mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ and $q_I \notin Q_1 \cup Q_2$ hold, and Q (Δ, $\mathcal{F}$) is defined from Q_1 and Q_2 (Δ_1 and Δ_2, $\mathcal{F}_1$ and $\mathcal{F}_2$ and $\mathcal{F}'$) as specified in* Proposition 5.5.1*. For all $w \in (2^{AP})^\omega$,*

$\mathcal{A}$ fin-accepts w iff there exists an index $0 \le i < \omega$ such that $\mathcal{A}_2$ fin-accepts w^i, and for all $0 \le j < i$, $\mathcal{A}_1$ fin-accepts w^j
or
$\mathcal{F}' = \emptyset$, and $\mathcal{A}_1$ fin-accepts w^i for all $0 \le i < \omega$.

Proof: (Only if) Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$, and let $G = \langle V, E, L\rangle$ be a fin-accepting run of $\mathcal{A}$ on w. By Proposition 2.3.7, there exists an index $0 \le i \le \omega$ and a chain of edges $(e_j)_{0 \le j < i+1}$, $e_j \in E \cap (V_j \times 2^{V_{j+1}})$, such that the transition $L(e_j) \in \Delta$ is an initial self-loop of $\mathcal{A}$ for all $0 \le j < i$, and if $i < \omega$ holds, then $L(e_i) \in \Delta$ is an initial transition of $\mathcal{A}$ that is not a self-loop.

We first show that if $w^j \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds for some $0 \leq j < i+1$, then $w^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds for all $0 \leq k \leq j$. Let $0 \leq j < i+1$ be an index such that $w^j \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds. The implication holds trivially if $j = 0$. Assume that $j \geq 1$ holds, and $w^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds for some $1 \leq k \leq j$. Because $k-1 \leq j-1 < i$ holds, the transition $t \overset{\text{def}}{=} L(e_{k-1}) = \langle q_I, \theta, F, Q' \rangle \in \Delta$ is an initial self-loop of $\mathcal{A}$, and because G is a fin-accepting run of $\mathcal{A}$, $w(k-1) \models \theta$ holds, and $w^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ holds for all $q' \in Q'$ (Proposition 2.3.9). Because t is an initial self-loop of $\mathcal{A}$, it follows from the definition of $\mathcal{A}$ that the automaton $\mathcal{A}_1$ has an initial transition $\langle q_{I1}, \theta, F_1, Q'_1 \rangle \in \Delta_1$ for some $F_1 \subseteq \mathcal{F}_1$ and $Q'_1 \subseteq Q_1$ such that $Q' = \big(Q'_1 \setminus \{q_{I1}\}\big) \cup \{q_I\}$ holds. Because $w^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q_{I1}})$ (Proposition 2.3.12) holds by the assumption, it follows that, for all $q' \in Q'_1$, $w^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q',\mathcal{F}_1}) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q'})$ (Lemma 5.5.2) holds, and because $w(k-1) \models \theta$ holds, it follows by Proposition 2.3.15 that $w^{k-1} \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds. By induction on decreasing values of k, it follows that $w^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds for all $0 \leq k \leq j$.

<u>$(i < \omega)$</u> If $i < \omega$ holds, then the transition $L(e_i) = \langle q_I, \theta, F, Q' \rangle \in \Delta$ is not an initial self-loop of $\mathcal{A}$. By the definition of $\mathcal{A}$, $L(e_i)$ corresponds to an initial transition $\langle q_{I2}, \theta, F_2, Q' \rangle \in \Delta_2$ of $\mathcal{A}_2$ for some $F_2 \subseteq \mathcal{F}_2$, and $Q' \subseteq Q_2$ holds. Again, because G is a fin-accepting run of $\mathcal{A}$, $w(i) \models \theta$ holds, and for all $q' \in Q'$, $w^{i+1} \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q',\mathcal{F}_2}) = \mathcal{L}_{\text{fin}}(\mathcal{A}_2^{q'})$ (Lemma 5.5.2) holds. By Proposition 2.3.15, it follows that $w^i \in \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ holds. Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds by assumption, also $\mathcal{A}_1$ fin-accepts w^i. By the above inductive argument, $w^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds for all $0 \leq k \leq i$, and the result follows.

<u>$(i = \omega)$</u> If $i = \omega$ holds, then $L(e_j)$ is an initial self-loop of $\mathcal{A}$ for all $0 \leq j < \omega$. Then necessarily $\mathcal{F}' = \emptyset$ holds, since otherwise $\text{fin}\big((e_j)_{0 \leq j < \omega}\big) \neq \emptyset$ would hold, contrary to the assumption that G is fin-accepting. Because each initial self-loop of $\mathcal{A}$ corresponds to an initial transition of $\mathcal{A}_1$, then the result follows immediately by the above inductive argument if the initial transition corresponding to $L(e_j)$ is not an initial self-loop of $\mathcal{A}_1$ for infinitely many $0 \leq j < \omega$ (if $L(e_j)$ corresponds to an initial transition of $\mathcal{A}_1$ that is not a self-loop of $\mathcal{A}_1$, it is straightforward to check that the index j can be used as the base case for the induction).

Otherwise there exists an index $0 \leq j < \omega$ such that the transition $L(e_{j+k})$ (which is an initial self-loop of $\mathcal{A}$) corresponds to an initial self-loop of $\mathcal{A}_1$ for all $0 \leq k < \omega$. Let $e_j = \langle \hat{v}_0, \widehat{V}' \rangle$ for some $\hat{v}_0 \in V_j$ and $\widehat{V}' \subseteq V_{j+1}$, and let $L(e_{j+k}) = \langle q_I, \hat{\theta}_k, \widehat{F}_k, \widehat{Q}'_k \rangle$, where $w(j+k) \models \hat{\theta}_k$ holds for all $0 \leq k < \omega$. Write $\widehat{Q}'_k \setminus \{q_I\} = \{q_{k,1}, q_{k,2}, \ldots, q_{k,n_k}\}$ $(0 \leq n_k < \omega)$ for all $0 \leq k < \omega$.

Define the graph $G' = \langle V', E', L' \rangle$, where

- $V'_0 \overset{\text{def}}{=} \{\hat{v}_0\}$, $V'_{k+1} \overset{\text{def}}{=} \{\hat{v}_{k+1}, v_{k,1}, \ldots, v_{k,n_k}\}$ for all $0 \leq k < \omega$,
- $E' \overset{\text{def}}{=} \bigcup_{0 \leq k < \omega} \big\{\langle \hat{v}_k, V'_{k+1} \rangle\big\}$,
- $L'(\hat{v}_k) \overset{\text{def}}{=} q_{I1}$, $L'(v_{k,\ell}) \overset{\text{def}}{=} q_{k,\ell}$ for all $0 \leq k < \omega$ and $1 \leq \ell \leq n_k$, and $L'\big(\langle \hat{v}_k, V'_{k+1} \rangle\big) \overset{\text{def}}{=} \big\langle q_{I1}, \hat{\theta}_k, \widehat{F}_k, (\widehat{Q}'_k \setminus \{q_I\}) \cup \{q_{I1}\} \big\rangle$ for all $0 \leq k < \omega$.

G' is a fin-accepting semi-run of $\mathcal{A}_1$ on w^j:

(Partitioning) $V'_0 = \{\hat{v}_0\}$ is a singleton, and V' is partitioned into disjoint finite levels (with edges only between successive levels).

(Causality) Let $v \in V'_k$ for some $0 \le k < \omega$. Then v either has no outgoing edges, or $v = \hat{v}_k$, in which case v has the unique outgoing edge $\langle v, V'_{k+1}\rangle \in E'$. On the other hand, each node $v \in V'_k$ ($1 \le k < \omega$) is a successor of the node $\hat{v}_{k-1} \in V'_{k-1}$ in G'. It follows that G' satisfies the forward semi-causality and backward causality constraints.

(Consistency of L') Clearly, $L'(\hat{v}_0) = q_{I1}$ holds. Let $e = \langle \hat{v}_k, V'_{k+1}\rangle \in E$ for some $0 \le k < \omega$. Because the transition $L(e_{j+k}) = \langle q_I, \hat{\theta}_k, \widehat{F}_k, \widehat{Q}'_k\rangle$ is a self-loop of $\mathcal{A}$ that corresponds to an initial self-loop of $\mathcal{A}_1$ and $\mathcal{F}' = \emptyset$ holds, it follows from the definition of $\mathcal{A}$ that Δ_1 contains the transition $\big\langle q_{I1}, \hat{\theta}_k, \widehat{F}_k, (\widehat{Q}'_k \setminus \{q_I\}) \cup \{q_{I1}\}\big\rangle = L'(e) = \big\langle L'(\hat{v}_k), \hat{\theta}_k, \widehat{F}_k, L(V'_{k+1})\big\rangle$. Furthermore, because G is a run of $\mathcal{A}$, $w(j+k) = w^j(k) \models \hat{\theta}_k$ holds, and thus the labeling L' is consistent.

(Acceptance) Clearly, $\mathcal{B}(G')$ contains the unique infinite branch $\beta \stackrel{def}{=} (e'_k)_{0\le k<\omega} = \big(\langle \hat{v}_k, V'_{k+1}\rangle\big)_{0\le k<\omega}$. Because the transition $L'(e'_k)$ inherits its acceptance conditions from the transition $L(e_{j+k})$ for all $0 \le k < \omega$, it follows that $\text{fin}(\beta) = \text{fin}\big((e'_k)_{0\le k<\omega}\big) = \text{fin}\big((e_{j+k})_{0\le k<\omega}\big) = \emptyset$, because G is a fin-accepting run of $\mathcal{A}$ on w. It follows that G' is a fin-accepting semi-run of $\mathcal{A}_1$ on w^j.

Let $v \in V'$ be a node with no outgoing edges in G'. Then $v = v_{k,\ell}$ holds for some $0 \le k < \omega$, $1 \le \ell \le n_k$, and $L'(v) = q_{k,\ell} \in \widehat{Q}'_k \setminus \{q_I\} \subseteq Q_1$. Because G is a fin-accepting run of $\mathcal{A}$ on w and $e_{j+k} \in E$ and $L(e_{j+k}) = \langle q_I, \hat{\theta}_k, \widehat{F}_k, \widehat{Q}'_k\rangle$ hold, it follows that $w(j+k) \models \hat{\theta}_k$ holds, and $\mathcal{A}^{q'}$ fin-accepts $w^{j+k+1} = (w^j)^{k+1}$ for all $q' \in \widehat{Q}'_k$. In particular, $(w^j)^{k+1} \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_{k,\ell}}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_{k,\ell},\mathcal{F}_1}) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q_{k,\ell}})$ holds, and because v is arbitrary, G' can be extended into a fin-accepting run of $\mathcal{A}_1$ on w^j by Proposition 2.3.14.

Since each level of the fin-accepting run G' includes a node labeled with the initial state of $\mathcal{A}_1$, it follows by Proposition 2.3.9 that $\mathcal{A}_1$ fin-accepts $(w^j)^k = w^{j+k}$ for all $0 \le k < \omega$. By the inductive argument given in the beginning of the proof, $\mathcal{A}_1$ fin-accepts w^k also for all $0 \le k \le j$, and thus $\mathcal{A}_1$ fin-accepts w^k for all $0 \le k < \omega$.

(If) The result follows in this direction by an obvious modification of the proof of the corresponding direction in Lemma 3.3.1 (when defining nonempty levels of a run of $\mathcal{A}$ on w, do not include nodes labeled with the state q_{I1} to any level with index less than $i+1$ to make it possible to label the chain of edges that emerges with initial transitions of $\mathcal{A}$ in a consistent way). $\square$

To complete the proof of Proposition 5.5.1, we show that the translation rule defined in the proposition preserves acceptance closure and the existence of representative states for acceptance conditions.

Lemma 5.5.4 *Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}$ be three alternating automata as specified in Proposition 5.5.1. The automaton $\mathcal{A}$ is an acceptance closed alternating automaton such that if $\mathcal{A}$ has an f-state for some $f \in \mathcal{F}$, then $\mathcal{A}$ has an f-representative state that is f-representative also in all automata $\mathcal{A}_n$ ($n \in \{1,2\}$) that have an f-state.*

Proof: ($\mathcal{A}$ is acceptance closed) Because $\mathcal{A}_1$ and $\mathcal{A}_2$ are acceptance closed automata, it is easy to see that every transition $t \in \Delta_1 \cup \Delta_2$ is f-closed for all $f \in \mathcal{F}$. Clearly, also every initial transition of $\mathcal{A}$ is f-closed for all $f \in \mathcal{F}$: if $\mathcal{A}$ has an initial f-transition, then this transition is a self-loop of $\mathcal{A}$ and thus includes the f-state q_I in its target states.

($\mathcal{A}$ has representative states for acceptance conditions) Let $f \in \mathcal{F}$ be an acceptance condition, and let $q \in Q = Q_1 \cup Q_2 \cup \{q_I\}$ be an f-state of $\mathcal{A}$. If $f \notin \mathcal{F}_1 \cup \mathcal{F}_2$ holds, then it follows from the definition of $\mathcal{A}$ that $q = q_I$ holds. Therefore, $\mathcal{A}$ has a unique f-state in this case, and this state is trivially an f-representative state of $\mathcal{A}$.

Otherwise, if $f \in \mathcal{F}_1 \cup \mathcal{F}_2$ holds, then $q \in Q_n$ holds for some $n \in \{1, 2\}$, or $q = q_I$, and the initial state q_{I1} of $\mathcal{A}_1$ is an f-state (otherwise q_I would not be an f-state). In either case, one of the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ contains an f-state, and it follows from the assumptions in Proposition 5.5.1 that this automaton contains an f-representative state q_f. (If both $\mathcal{A}_1$ and $\mathcal{A}_2$ contain an f-state, we may assume that the state q_f is f-representative in both automata.) We show that q_f is an f-representative state of $\mathcal{A}$.

Let $q \in Q$ be an f-state of $\mathcal{A}$, and let $w \in (2^{AP})^\omega$. If $q \in Q_n$ holds for some $n \in \{1, 2\}$, then it is straightforward to check that the state q_f satisfies the two conditions required of an f-representative state with respect to the f-state q (Sect. 4.3.3), because q_f is an f-representative state of $\mathcal{A}_n$, and $\mathcal{A}^{q'}$ fin-accepts w (by avoiding an initial f-transition) iff $\mathcal{A}^{q',\mathcal{F}_n}$ ($= \mathcal{A}_n^{q'}$, Lemma 5.5.2) fin-accepts w (by avoiding an initial f-transition) for all $q' \in Q_n$.

Otherwise, if $q = q_I$ holds, it follows that the initial state q_{I1} of $\mathcal{A}_1$ is an f-state, and $q_f \in Q_1$ is an f-representative state of $\mathcal{A}_1$. We check that the properties required of an f-representative state of $\mathcal{A}$ hold for q_f also in this case.

If $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ holds, then, because the automata $\mathcal{A}^{q_I}$ and $\mathcal{A}$ fin-recognize the same language (Proposition 2.3.12), $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds, and it follows by Lemma 5.5.3 that there exists an index $0 \leq i \leq \omega$ such that $w^j \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ ($= \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q_{I1}})$) holds for all $0 \leq j < i$, and if $i < \omega$ holds, then $w^i \in \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ ($= \mathcal{L}_{\text{fin}}(\mathcal{A}_2^{q_{I2}})$) holds.

($w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ implies $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_I})$) Suppose that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ holds, and let $0 \leq i \leq \omega$ be the index defined above.

If $i \geq 1$ holds, then $w^0 = w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q_{I1}})$ holds, and it is easy to see that also $w^1 \in \mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ holds. Because $q_f \in Q_1$ holds, $\mathcal{A}^{q_f}$ fin-accepts w by avoiding an initial f-transition iff $\mathcal{A}^{q_f,\mathcal{F}_1}$ ($= \mathcal{A}_1^{q_f}$, Lemma 5.5.2) fin-accepts w by avoiding an initial f-transition, and it follows that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}_1^{q_f}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q_{I1}})$ holds. Because q_f is an f-representative state of $\mathcal{A}_1$, $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}_1^{q_{I1}})$ holds. By Proposition 2.3.15, the automaton $\mathcal{A}_1$ has an initial transition $\langle q_{I1}, \theta, F, Q' \rangle \in \Delta_1$ such that $w(0) \models \theta$ and $f \notin F$ hold, and $w^1 \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1^q) = \mathcal{L}_{\text{fin}}(\mathcal{A}_1^{q,\mathcal{F}}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds for all $q \in Q'$. By the definition of $\mathcal{A}$, $\mathcal{A}$ has the initial transition $\big\langle q_I, \theta, F, (Q' \setminus \{q_{I1}\}) \cup \{q_I\} \big\rangle \in \Delta$. It now follows that $\mathcal{A}^q$ fin-accepts w^1 for all $q \in (Q' \setminus \{q_{I1}\}) \cup \{q_I\}$, and because $w(0) \models \theta$ and $f \notin F$ hold, it follows by Proposition 2.3.15 that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}) = \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_I})$ holds.

If $i = 0$, then $w^0 = w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ holds. By Proposition 2.3.15, the automaton $\mathcal{A}_2$ has an initial transition $\langle q_{I2}, \theta, F, Q' \rangle \in \Delta_2$ such that $w(0) \models \theta$ holds, and $\mathcal{A}_2^q$ fin-accepts w^1 for all $q \in Q'$. In this case the automaton $\mathcal{A}$ has the initial transition $\langle q_I, \theta, \emptyset, Q' \rangle \in \Delta$, and it is easy to see that $w^1 \in \mathcal{L}_{\text{fin}}(\mathcal{A}_2^q) = \mathcal{L}_{\text{fin}}(\mathcal{A}_2^{q,\mathcal{F}}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds for all $q \in Q'$. By Proposition 2.3.15, it follows that $\mathcal{A}$ fin-accepts w by avoiding an initial f-transition, and thus $w \in \mathcal{L}_{\text{fin}}^f(\mathcal{A}) = \mathcal{L}_{\text{fin}}^f(\mathcal{A}^{q_I})$ holds also in this case.

<u>*($w \in \mathcal{L}_{\text{fin}}^f(\mathcal{A}^{q_I})$ implies $w^j \in \mathcal{L}_{\text{fin}}^f(\mathcal{A}^{q_f})$ for some $0 \leq j < \omega$)*</u> Assume that $w \in \mathcal{L}_{\text{fin}}^f(\mathcal{A}^{q_I})$ holds, and let $0 \leq i \leq \omega$ be the index defined above. Clearly, $w^0 \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds if $i \geq 1$. Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds by the assumptions in Proposition 5.5.1, however, $w^0 \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ actually holds also if $i = 0$ (in this case $w^0 \in \mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ holds). It follows that the automaton $\mathcal{A}_1$ fin-accepts w. Similarly to Lemma 5.4.8, it is straightforward to show (by applying Proposition 2.3.7 to a fin-accepting run of $\mathcal{A}_1$ on w) that there exists an index $0 \leq k < \omega$ such that $w^k \in \mathcal{L}_{\text{fin}}^f(\mathcal{A}_1)$ holds, and because the state q_f is an f-representative state of $\mathcal{A}_1$, it follows that $(w^k)^\ell = w^{k+\ell} \in \mathcal{L}_{\text{fin}}^f(\mathcal{A}_1^{q_f}) = \mathcal{L}_{\text{fin}}^f(\mathcal{A}^{q_f})$ holds for some $0 \leq \ell < \omega$.

We conclude that q_f is an f-representative state of $\mathcal{A}$ that is f-representative also in every automaton $\mathcal{A}_n$ ($n \in \{1, 2\}$) that has an f-state. Because the same holds for all acceptance conditions $f \in \mathcal{F}_1 \cup \mathcal{F}_2$, the result follows. $\square$

Proposition 5.5.1 can be used directly in the proof of Theorem 3.3.2 to show that the procedure for translating LTL formulas into alternating automata remains correct if the set of translation rules is extended with special rules for the U_s and U_w connectives shown in the upper half of Table 5.3 (see also Fig. 5.6 (b) and Fig. 5.6 (c)). Furthermore, because the automata built using the rules are acceptance closed automata with representative states for all of their acceptance conditions f for which they have an f-state, the translation procedure extended with the new rules yields automata which can be translated into nondeterministic automata without introducing new acceptance conditions (Corollary 4.3.7).

The Release Connectives

The rules for the R_s and R_w connectives can again be obtained by combining the (original) $\wedge$ rule with the new rules for the corresponding Until connectives. In particular, due to the identities

$$(\varphi_1 \,\mathsf{R}_\mathsf{s}\, \varphi_2) \equiv \big((\varphi_2 \,\mathsf{U}_\mathsf{s}\, (\varphi_1 \wedge \varphi_2)\big) \quad \text{and} \quad (\varphi_1 \,\mathsf{R}_\mathsf{w}\, \varphi_2) \equiv \big((\varphi_2 \,\mathsf{U}_\mathsf{w}\, (\varphi_1 \wedge \varphi_2)\big)$$

and the fact that $\mathcal{L}\big((\varphi_1 \wedge \varphi_2)\big) = \mathcal{L}(\varphi_1) \cap \mathcal{L}(\varphi_2) \subseteq \mathcal{L}(\varphi_2)$ always holds for all pairs of LTL formulas $\varphi_1, \varphi_2 \in LTL(AP)$, it follows that the language containment assumption in Proposition 5.5.1 holds trivially between the top-level subformulas of the Until formulas corresponding to the Release formulas. This fact makes it possible to apply the new Until rules in the derivation of the Release rules, and thus the original translation rules for the Release connectives can be replaced with ones that allow a slightly simplified transition structure for the compound automaton.

Finally, the identities of Table 5.1 allow a further improvement in the translation of Release formulas of the form $(\varphi_1 \,\mathsf{R}\, \varphi_2)$, this time under the

Table 5.3: Refined translation rules for the binary temporal connectives (continuation of Table 3.1)

$\circ$	$\mathcal{F}_\circ$	$\Delta_\circ$
$\mathsf{U_s}$	$\{f\}$	$\Big\{\langle q_I,\theta_1,\{f\},(Q'_1\setminus\{q_{I1}\})\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1\Big\} \cup \Big\{\langle q_I,\theta_2,\emptyset,Q'_2\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\}$ $\boxed{\textit{if } \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)\subseteq\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)}$
		$\Big\{\langle q_I,\theta_1,\{f\},Q'_1\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1\Big\} \cup \Big\{\langle q_I,\theta_2,\emptyset,Q'_2\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\}$ $\boxed{\textit{otherwise}}$
$\mathsf{U_w}$	$\emptyset$	$\Big\{\langle q_I,\theta_1,F_1,(Q'_1\setminus\{q_{I1}\})\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1\Big\} \cup \Big\{\langle q_I,\theta_2,\emptyset,Q'_2\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\}$ $\boxed{\textit{if } \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)\subseteq\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)}$
		$\Big\{\langle q_I,\theta_1,\emptyset,Q'_1\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1\Big\} \cup \Big\{\langle q_I,\theta_2,\emptyset,Q'_2\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\}$ $\boxed{\textit{otherwise}}$
$\mathsf{R_s}$	$\{f\}$	$\Big\{\langle q_I,\theta_2,\{f\},(Q'_2\setminus\{q_{I2}\})\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\} \cup \Big\{\langle q_I,\theta_1,\emptyset,Q'_1\rangle \;\Big\vert\; \langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1\Big\}$ $\boxed{\textit{if } \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)\subseteq\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)}$
		$\Big\{\langle q_I,\theta_2,\{f\},(Q'_2\setminus\{q_{I2}\})\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\} \cup \left\{\langle q_I,(\theta_1\wedge\theta_2),\emptyset,Q'_1\cup Q'_2\rangle \;\middle\vert\; \begin{matrix}\langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1,\\ \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\end{matrix}\right\}$ $\boxed{\textit{otherwise}}$
$\mathsf{R_w}$	$\emptyset$	$\Big\{\langle q_I,\theta_2,F_2,(Q'_2\setminus\{q_{I2}\})\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\} \cup \Big\{\langle q_I,\theta_1,\emptyset,Q'_1\rangle \;\Big\vert\; \langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1\Big\}$ $\boxed{\textit{if } \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)\subseteq\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)}$
		$\Big\{\langle q_I,\theta_2,F_2,(Q'_2\setminus\{q_{I2}\})\cup\{q_I\}\rangle \;\Big\vert\; \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\Big\} \cup \left\{\langle q_I,(\theta_1\wedge\theta_2),\emptyset,Q'_1\cup Q'_2\rangle \;\middle\vert\; \begin{matrix}\langle q_{I1},\theta_1,F_1,Q'_1\rangle\in\Delta_1,\\ \langle q_{I2},\theta_2,F_2,Q'_2\rangle\in\Delta_2\end{matrix}\right\}$ $\boxed{\textit{otherwise}}$

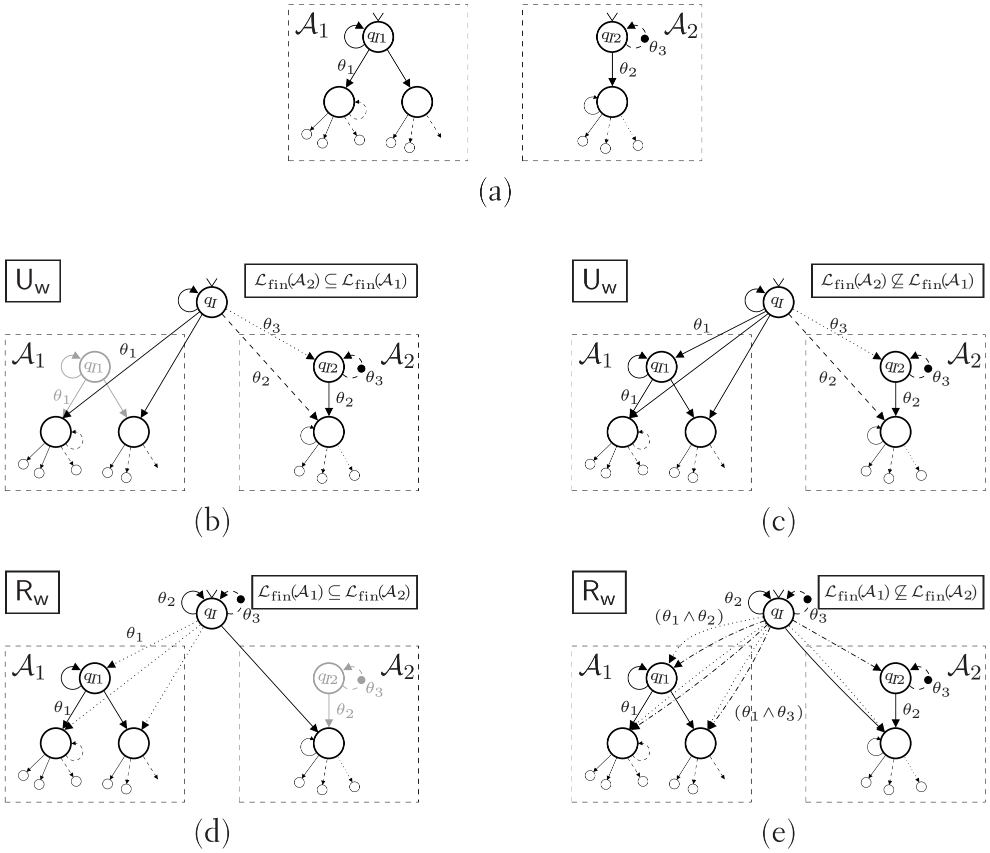


Fig. 5.6: Automata built using the refined translation rules for the weak binary temporal connectives. (a) Two component automata $\mathcal{A}_1$ and $\mathcal{A}_2$; (b) Automaton built from $\mathcal{A}_1$ and $\mathcal{A}_2$ with the $\mathsf{U_w}$ translation rule under the assumption $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)$; (c) Automaton built from the component automata using the $\mathsf{U_w}$ rule without the language containment assumption; (d) Automaton built from $\mathcal{A}_1$ and $\mathcal{A}_2$ with the $\mathsf{R_w}$ translation rule under the assumption $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1) \subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$; (e) Automaton built from the component automata using the $\mathsf{R_w}$ rule without the assumption. The initial transitions of the automata built for the corresponding strong connectives have the same target states as the initial transitions of the above automata; however, all the initial self-loops of the automata for the strong connectives share an acceptance condition that is not included in either of the component automata

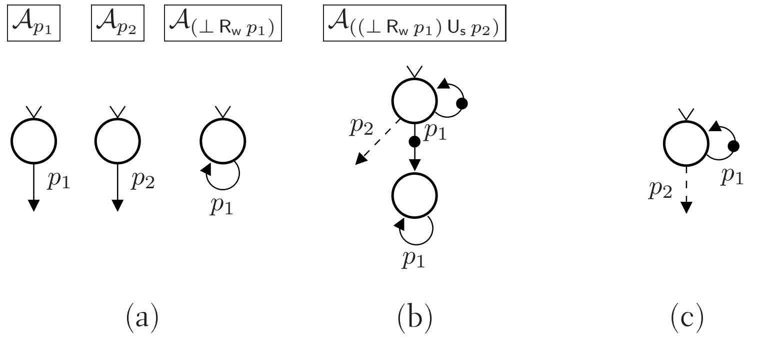


Fig. 5.7: The refined Until translation rules cannot be universally substituted for the original ones. (a) Automata $\mathcal{A}_{p_1}$, $\mathcal{A}_{p_2}$ and $\mathcal{A}_{(\bot\,\mathsf{R_w}\,p_1)}$ built for the formulas p_1, p_2, and $(\bot\,\mathsf{R_w}\,p_1)$, respectively; (b) Automaton built for the formula $\big((\bot\,\mathsf{R_w}\,p_1)\,\mathsf{U_s}\,p_2\big)$ from $\mathcal{A}_{(\bot\,\mathsf{R_w}\,p_1)}$ and $\mathcal{A}_{p_2}$ with the original translation rules; (c) Automaton built from the same automata with the refined translation rules, ignoring the requirement on the language containment relationship between the languages fin-accepted by these automata

language containment relationship $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$. Because $(\varphi_1\,\mathsf{R_s}\,\varphi_2) \equiv (\varphi_2\,\mathsf{U_s}\,\varphi_1)$ and $(\varphi_1\,\mathsf{R_w}\,\varphi_2) \equiv (\varphi_2\,\mathsf{U_w}\,\varphi_1)$ hold in this case, and because the assumption that $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$ holds implies the language containment assumption in Proposition 5.5.1 for the Until formulas, the translation of the Release formulas reduces in this case to the translation of Until formulas using the new translation rules. (As noted in Sect. 5.3, we may not need to apply a rule explicitly if we are able to reuse an automaton built for an Until formula.) Using an Until rule for the translation in this case removes the need to collect all pairs of initial transitions of the component automata corresponding to the subformulas φ_1 and φ_2, which reduces the worst-case number of initial transitions in the compound automaton. The new rules for the Release connectives are shown in the lower half of Table 5.3; see also Fig. 5.6 (d) and Fig. 5.6 (e) for illustration.

We end this section with an example to show that the language containment assumption used in the new translation rules for the $\mathsf{U_s}$ or $\mathsf{U_w}$ connectives cannot be lifted in the general case.

Example 5.5.5 Figure 5.7 shows two automata built for the LTL formula

$$\big((\mathsf{G}p_1)\,\mathsf{U_s}\,p_2\big) \equiv \big((\bot\,\mathsf{R_w}\,p_1)\,\mathsf{U_s}\,p_2\big) \in LTL\big(\{p_1, p_2\}\big),$$

where the automaton in Fig. 5.7 (b) is obtained from the formula using the original translation rules (with the usual restriction of the automaton to states reachable from its initial state), whereas the automaton in Fig. 5.7 (c) is (erroneously) built from the formula by applying the new rules, ignoring the precondition on the language containment relationship between its top-level subformulas $(\bot\,\mathsf{R_w}\,p_1)$ and p_2; clearly, $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_{p_2})\ (= \mathcal{L}(p_2)) \not\subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_{(\bot\,\mathsf{R_w}\,p_1)})$ $(= \mathcal{L}(\mathsf{G}p_1))$ holds in this case. It is easy to see that the automaton shown in Fig. 5.7 (c) fin-accepts the word $\{p_1\}\{p_2\}^\omega$, which is, however, not a model of the LTL formula $\big((\mathsf{G}p_1)\,\mathsf{U_s}\,p_2\big)$. ■

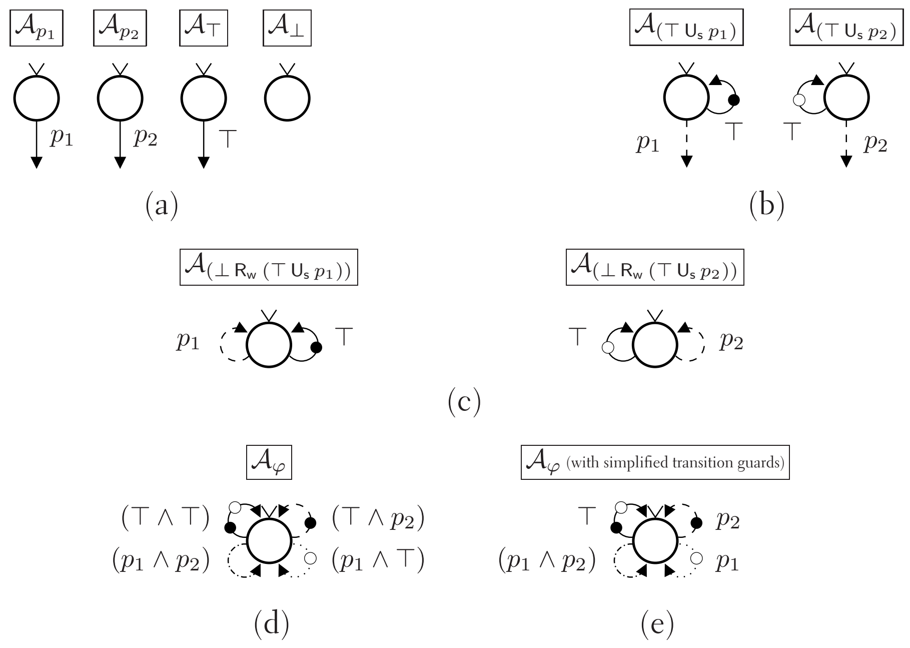


Fig. 5.8: Building an automaton for the LTL formula $\varphi \overset{def}{\equiv} \Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big) \land \big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big)$ using the refined translation rules. (a) Automata for the atomic subformulas of φ; (b) Automata for the formulas $(\top\,\mathsf{U_s}\,p_1)$ and $(\top\,\mathsf{U_s}\,p_2)$; (c) Automata for the formulas $\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big)$ and $\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)$; (d) Automaton for the formula φ; (e) The automaton obtained from (e) via transition guard simplification

5.6 DISCUSSION

In this section we illustrate and discuss some effects of adding the new translation rules to the basic procedure for translating LTL formulas into alternating automata.

5.6.1 Translation Example Revisited

We start with an example to illustrate the behavior of the new translation rules.

Example 5.6.1 Consider again the LTL formula

$$(\varphi \lor \psi) \overset{def}{\equiv} \Big[\Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big) \land \big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big) \lor \big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)\Big]$$

from Ex. 3.1.1; as in Ex. 3.1.1, we translate the formula into an automaton by dealing with its top-level subformulas φ and ψ separately.

The basic rules for the atomic subformulas of φ yield again the automata shown in Fig. 5.8 (a). Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_{p_1}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_{\top})$ and $\mathcal{L}_{\text{fin}}(\mathcal{A}_{p_2}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_{\top})$ obviously hold, we can apply the new rule for the $\mathsf{U_s}$ connective to build the automata for the formulas $(\top\,\mathsf{U_s}\,p_1)$ and $(\top\,\mathsf{U_s}\,p_2)$ as shown in Fig. 5.8 (b). (Actually, because the automaton $\mathcal{A}_{\top}$ has no initial self-loops, we obtain in this case the same compound automata as before.)

Because $\mathcal{L}_{\text{fin}}(\bot) = \emptyset$ holds, the language fin-recognized by $\mathcal{A}_{\bot}$ is trivially a subset of the language fin-recognized by any automaton. Therefore, by

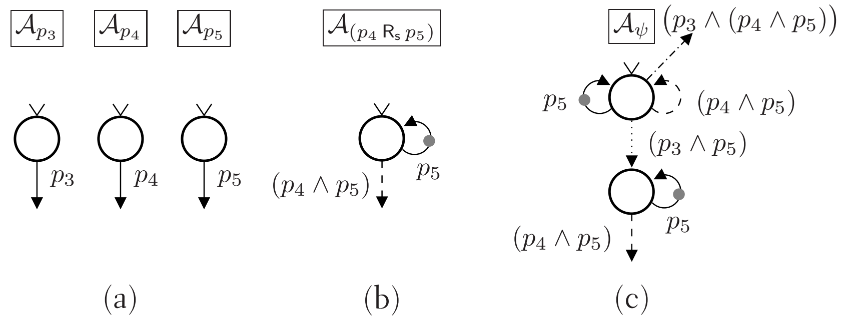


Fig. 5.9: Building an automaton for the formula $\psi \overset{def}{\equiv} \big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)$ using the refined translation rules. (a) Automata for the atomic subformulas of ψ; (b) Automaton for the formula $(p_4 \,\mathsf{R_s}\, p_5)$; (c) Automaton for the formula ψ

using the appropriate $\mathsf{R_w}$ rule that makes use of this language containment assumption, we obtain the automata shown in Fig. 5.8 (c) for the subformulas $\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big)$ and $\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big)$. We then apply the new rule for the $\land$ connective (Table 5.2) to build an automaton for the formula φ. As shown in Fig. 5.8 (d), merging all pairs of initial self-loops of the component automata yields a single-state automaton for the formula φ instead of the five-state automaton (Fig. 3.2 (d), p. 46) built using the basic translation rules. Figure 5.8 (e) shows the same automaton after simplifying its transition guards as described in Sect. 5.1.2.

We then repeat the translation for the formula ψ. As before, we start from the automata for the atomic formulas (Fig. 5.9 (a)). Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_{p_4}) \not\subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_{p_5})$ holds, we use the universally applicable version of the optimized translation rule for the $\mathsf{R_s}$ connective to define an automaton for the formula $(p_4 \,\mathsf{R_s}\, p_5)$ (Fig. 5.9 (b)). The corresponding rule for the $\mathsf{R_w}$ connective is applied to build an automaton for the formula $\big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)$ (Fig. 5.9 (c)). In comparison to the automaton obtained in Ex. 3.1.1 (Fig. 3.3 (c), p. 46), the refined translation rules allow us to reduce the number of target states in one transition of the automaton.

Finally, we build an automaton for the formula

$$\Big[\Big(\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big) \land \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big)\Big) \lor \big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)\Big]$$

by applying the $\lor$ rule as shown in Fig. 5.10. This automaton has three states less than the automaton built for the same formula in Ex. 3.1.1 (cf. Fig. 3.4, p. 47). Additionally, no transition in the automaton built using the refined rules has more than one target state, which is not the case for the automaton obtained using the basic translation rules. ■

5.6.2 Comparison of the Basic and the Refined Translation Rules

The refined translation rules behave very much like the basic rules given in Sect. 3.1. For example, because each new translation rule takes a new initial state for the automaton to be built, the correspondence between states in the automaton and node subformulas of a given formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$ is preserved, and because no new rule changes the transition structure of the

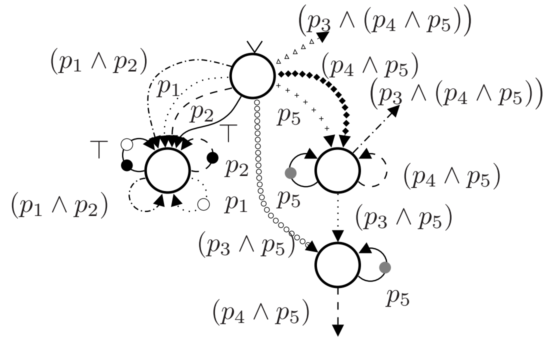


Fig. 5.10: Automaton constructed for the LTL formula $\Big[\Big(\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big) \wedge \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big)\Big) \vee \big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)\Big]$ using the refined translation rules

component automata to which the rule is applied, the compound automata will again be self-loop alternating automata. Also all transition guards are still formed as conjunctions of atomic formulas.

The main difference between the original and the new rules concerns the handling of acceptance conditions. Recall from the discussion in Sect. 3.1.1 that, in any automaton built using the original translation rules, every transition with a nonempty set of acceptance conditions always starts from a state corresponding to a strong temporal eventuality subformula. Furthermore, all self-loops starting from such a state share the same acceptance condition that is nevertheless unique in the sense that it is never included in the acceptance conditions of any transition starting from another state in the automaton. The refined rules, however, change this behavior: although the rules for the strong temporal connectives still introduce new acceptance conditions as before, the rules for the weak temporal connectives and the $\wedge$ connective may cause an initial transition of a compound automaton to inherit its acceptance conditions from a transition in a component automaton. Intuitively, an acceptance condition may thus propagate from transitions defined during the translation towards transitions defined later in the translation via chains of states corresponding to nested weak eventualities or conjunctions in the given LTL formula. The "oldest" state of each such chain corresponds to a strong temporal eventuality subformula or the conjunction of binary temporal formulas. Therefore, an automaton built using the new rules may contain self-loops that share a common acceptance condition even though they have different source states; additionally, as illustrated in Ex. 5.6.1, applications of the new rule for the $\wedge$ connective may result in states with self-loops associated with multiple acceptance conditions. Nevertheless, because the new rules preserve the acceptance closure of transitions and the existence of representative states for acceptance conditions (Lemma 5.4.8 and Lemma 5.5.4; as a matter of fact, a state corresponding to a strong temporal eventuality will remain representative for the corresponding acceptance condition throughout the translation), all automata built using the rules can still be translated into nondeterministic automata using the simple construction of Theorem 4.3.2 (or the construction of Proposition 4.5.2, which exploits syntactic implications).

Clearly, changing the translation rules necessitates a reconsideration of the upper bounds obtained in Sect. 3.2 for the sizes of the components of a self-loop alternating automaton corresponding to an LTL formula $\varphi \in LTL^{\mathrm{PNF}}(AP)$. Recall that the contribution of any automaton built for some subformula $\psi \in \mathrm{NSub}(\varphi)$ to the number of states in the automaton for φ depends on the number of states reachable from the initial state of the automaton for ψ; obviously, this fact still holds when using the new translation rules.

In the worst case, the original translation rules for the binary temporal connectives create compound automata in which both initial states of the component automata are reachable from the initial state of the compound automaton. (This worst case occurs whenever both component automata have initial self-loops.) Although the new translation rules sometimes avoid introducing transitions to the initial state of one of the component automata even if this automaton has an initial self-loop, it is easy to see that the worst case cannot be completely avoided since the rules for the Until connectives under a negative language containment assumption coincide with the original translation rules. Nevertheless, the number of states reachable from the initial state of an automaton built for a binary pure temporal formula using the new rules will never exceed the corresponding number of states in an automaton built for the same formula using the original rules, and thus the size limit of Corollary 3.2.2 remains valid for a translation procedure extended with the new rules for the binary temporal connectives.

The reasoning used in Sect. 3.2 to arrive at Proposition 3.2.1 does not apply, however, to the new translation rule for the $\wedge$ connective. Because this rule may introduce initial self-loops to a compound automaton, the initial state of this compound automaton may become reachable from itself; obviously, this never occurs when using the original translation rule for the $\wedge$ connective. Because of these initial self-loops, the initial state of the compound automaton will thus remain reachable also from any automaton obtained from this compound automaton using further translation rules. Consequently, the upper bound given in Corollary 3.2.2 for the size of an automaton corresponding to an LTL formula $\varphi \in LTL(AP)$ is not valid for a translation procedure extended with the new rule for the $\wedge$ connective; a straightforward correction to this result necessitates taking also all formulas of the form $(\varphi_1 \wedge \varphi_2) \in \mathrm{Sub}(\varphi)$ into account. This version of the result is, however, less optimal than the original one. As a matter of fact, it is easy to find examples where the original translation rule outperforms the new rule as far as the number of states reachable from the initial state of the final automaton is concerned.

Example 5.6.2 Consider translating the formula

$$\big((\mathsf{F}p_1 \wedge \mathsf{F}p_2) \vee p_3\big) \equiv \Big(\big((\top \,\mathsf{U_s}\, p_1) \wedge (\top \,\mathsf{U_s}\, p_2)\big) \vee p_3\Big) \in LTL\big(\{p_1, p_2, p_3\}\big)$$

into an automaton using both the original and the new translation rules. Applying the original $\wedge$ rule to the automata built for the formulas $(\top \,\mathsf{U_s}\, p_1)$ and $(\top \,\mathsf{U_s}\, p_2)$ (and then simplifying the guards of transitions) results in the automaton shown in Fig. 5.11 (a). On the other hand, using the new translation rule to build an automaton for the same formula will cause the self-loops

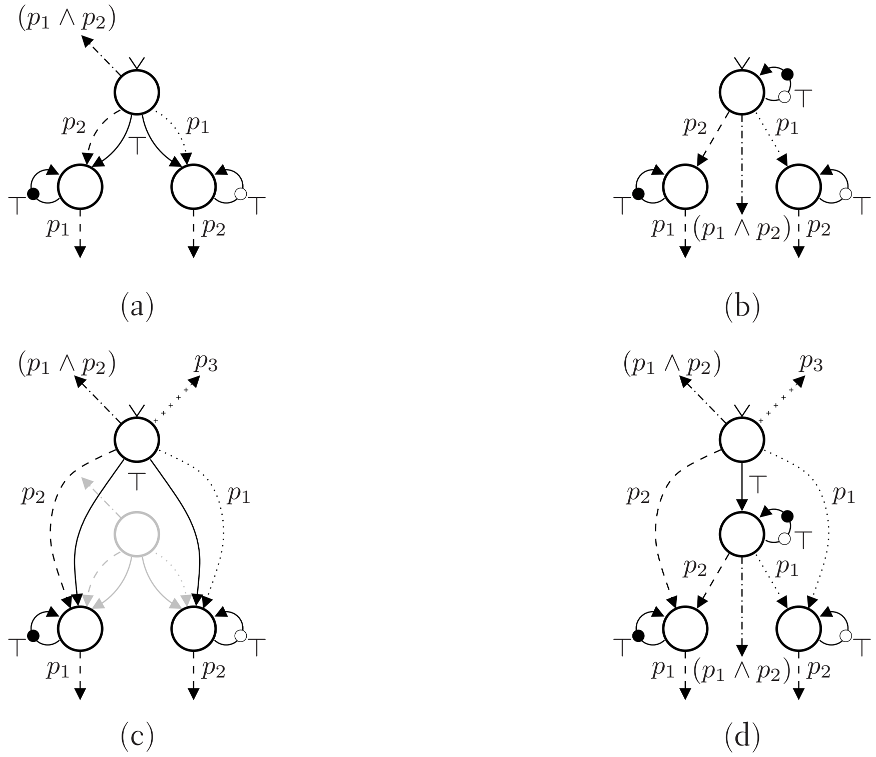


Fig. 5.11: Greedy application of the refined translation rule for the $\wedge$ connective may result in alternating automata with a suboptimal number of states. (a)–(b) Automata built for the formula $(\mathsf{F}p_1 \wedge \mathsf{F}p_2)$ using the original (a) and refined rules (b), respectively; (c)–(d) Automata built for the formula $\big((\mathsf{F}p_1 \wedge \mathsf{F}p_2) \vee p_3\big)$ by applying the $\vee$ translation rule to the automata (a) and (b), respectively

starting from the initial states of the component automata to be merged into an initial self-loop of the compound automaton (Fig. 5.11 (b)).

Applying the $\vee$ rule to the automaton built using the original rules results in the automaton shown in Fig. 5.11 (c). Clearly, the initial state of the automaton shown in Fig. 5.11 (a) is not reachable from the initial state of this automaton. On the other hand, applying the same rule to the automaton in Fig. 5.11 (b) results in "unrolling" the initial self-loop of this automaton into an initial transition of the automaton shown in Fig. 5.11 (d). However, there are now three (instead of two) states reachable from the initial state of this automaton. ■

As seen from the above example, the assumption that merging the initial segments of two branches of an accepting run of a compound automaton (built using the original $\wedge$ rule) results in a reduction in the number of states in the automaton is obviously very optimistic. To minimize the number of states in the alternating automaton, the rule should therefore be used only sparingly, for example, only if the application of the rule leaves an initial state of a component automaton unreachable from the initial state of the compound automaton built using the rule. For instance, this simple heuristic would already prevent the situation that arises in Ex. 5.6.2.

Nevertheless, there is a difference between the automata in Fig. 5.11 (c) and Fig. 5.11 (d) as regards translating these automata into nondeterministic automata: because the automaton in Fig. 5.11 (d) has no transitions with two or more target states, this automaton can be completed directly into a nondeterministic automaton by an application of Lemma 4.6.1. On the other hand, this completion result does not apply to the automaton shown in Fig. 5.11 (c); as a matter of fact, applying the universal subset construction (Theorem 4.3.2) to this alternating automaton yields a nondeterministic automaton that is identical to the one obtained from the automaton in Fig. 5.11 (d) as described above. Intuitively, the new rule for the $\wedge$ connective can thus be said to "reduce universality" (i.e., the number of target states) in the transitions of the alternating automaton at the cost of the size of the state set of the automaton.

5.6.3 Extension of the Subclass LTL^{CND}

We end this section by studying the effects of the new translation rules on the set of LTL formulas closed under translation into automata which can be completed into nondeterministic automata without applying the universal subset construction (previously considered in Sect. 4.6). The fact that the automata built from certain LTL formulas we have seen in previous examples (Fig. 5.10 and Fig. 5.11 (d)) can be completed into nondeterministic automata by an application of Lemma 4.6.1 is not a coincidence. First, we shall list closure properties of the new translation rules, and then use them to define a simple, yet more expressive, syntactic extension of the subclass LTL^{CND}, for which the satisfiability problem (discussed in Sect. 4.6.5) remains **NP**-complete.

Closure Properties of Refined Translation Rules
Let $\varphi \in LTL(AP)$ be an LTL formula, and let $\theta_1, \theta_2 \in PL(AP)$ be two Boolean formulas. It is clear from the syntactic definition of $LTL^{\mathrm{CND}}(AP)$ (Sect. 4.6.3, p. 107) that this subclass of LTL includes all formulas of the form θ_1, $(\theta_1 \,\mathsf{U}\, \theta_2)$ and $(\theta_1 \,\mathsf{R}\, \theta_2)$ (where both strong and weak variants of U and R are allowed). More precisely, as argued in Sect. 4.6.3, the basic translation rules map any formula of one of these forms into a self-loop automaton, in which the subautomaton rooted at the initial state of the automaton consists of a single state; obviously, this fact still holds when using the new translation rules for the $\wedge$, U and R connectives. In short, we say that these formulas translate into *single-state* automata; it is clear that every single-state automaton is trivially a self-loop automaton that can be extended into a nondeterministic automaton by applying Lemma 4.6.1. Furthermore, we say that an automaton is a *single-state loop* automaton iff the automaton is a single-state automaton, all initial transitions of which are self-loops. For illustration, see Fig. 5.8 (a) and Fig. 5.8 (b) for single-state automata, and Fig. 5.8 (c) and Fig. 5.8 (d) for single-state loop automata.

The new translation rules have the following simple closure properties on the translation of LTL formulas into single-state (loop) automata.

Lemma 5.6.3 *Let $\varphi \in LTL(AP)$ be an LTL formula which can be translated into a single-state automaton. The formulas $(\varphi \,\mathsf{U}\, \bot)$ and $(\bot \,\mathsf{R}\, \varphi)$ can be translated into single-state loop automata.*

Proof: Let $\mathcal{A}_\varphi$ and $\mathcal{A}_\bot$ be single-state automata built for the formulas φ and

$\perp$, respectively. Because $\mathcal{L}(\perp) = \emptyset \subseteq \mathcal{L}(\varphi)$ holds trivially, the formulas $(\varphi \,\mathsf{U}\, \perp)$ and $(\perp \mathsf{R}\, \varphi)$ can be translated into automata using the optimized translation rules in Table 5.3. Because the automaton built for the formula $\perp$ has no initial transitions, it is easy to see that every initial transition of the compound automaton built from $\mathcal{A}_\varphi$ and $\mathcal{A}_\perp$ will be a self-loop obtained from an initial transition of the automaton $\mathcal{A}_\varphi$. Because $\mathcal{A}_\varphi$ is a single-state automaton, the target state set of each transition of $\mathcal{A}_\varphi$ is either the empty set, or it consists of the initial state of $\mathcal{A}_\varphi$. The result now follows because the optimized translation rules remove the initial state of $\mathcal{A}_\varphi$ from every initial transition of $\mathcal{A}_\varphi$ when converting it to an initial self-loop of the compound automaton. □

Lemma 5.6.4 *Let $\varphi_1, \varphi_2 \in LTL(AP)$ be two LTL formulas translatable into single-state loop automata. The LTL formula $(\varphi_1 \wedge \varphi_2)$ can be translated into a single-state loop automaton.*

Proof: Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be single-state loop automata built for the formulas φ_1 and φ_2, respectively. Because all pairs of initial transitions of $\mathcal{A}_1$ and $\mathcal{A}_2$ consist of self-loops of the automata, all of these pairs of transitions are merged by the new rule for the $\wedge$ connective into initial self-loops of the compound automaton built from $\mathcal{A}_1$ and $\mathcal{A}_2$; furthermore, the initial states of the component automata are not included in the target states of these self-loops. The result now follows because $\mathcal{A}_1$ and $\mathcal{A}_2$ are single-state automata. □

Lemma 5.6.3 and Lemma 5.6.4 can be used to identify a simple syntactic subclass of LTL formulas which can be translated into single-state automata. Formally, we define this subclass $LTL^{\text{1-state}}(AP)$ (in terms of an auxiliary subclass $LTL^{\text{1-loop}}(AP)$) as the smallest set of LTL formulas closed under finite application of the mutually recursive rules

$$\begin{gathered}\text{If } \theta_1, \theta_2 \in PL(AP) \text{ and } \varphi \in LTL^{\text{1-loop}}(AP)\text{, then}\\ \theta_1, (\theta_1 \,\mathsf{U}\, \theta_2), (\theta_1 \,\mathsf{R}\, \theta_2), \varphi \in LTL^{\text{1-state}}(AP)\text{; and}\end{gathered}$$

$$\begin{gathered}\text{if } \varphi \in LTL^{\text{1-state}}(AP) \text{ and } \psi_1, \psi_2 \in LTL^{\text{1-loop}}(AP)\text{, then}\\ (\varphi \,\mathsf{U}\, \perp), (\perp \mathsf{R}\, \varphi), (\psi_1 \wedge \psi_2) \in LTL^{\text{1-loop}}(AP).\end{gathered}$$

(where U and R can be binary temporal connectives of any strength).

The above syntactic definition of $LTL^{\text{1-state}}(AP)$ was obtained without considering the semantics of the formulas formed using the recursive rules. As a matter of fact, there are in practice only few meaningful ways to apply the recursive rules to obtain formulas that cannot be trivially expressed as simpler formulas in the subclass. For example, given a formula $\varphi \in LTL^{\text{1-state}}(AP)$, it does not make sense in practice to write formulas of the form $(\varphi \,\mathsf{U}_\mathsf{s}\, \perp)$ or $(\perp \mathsf{R}_\mathsf{s}\, \varphi)$, since each of these formulas is trivially equivalent to the formula $\perp$ as is easily seen from the semantics of the strong binary temporal operators. Likewise, if $\varphi \in LTL^{\text{1-loop}}(AP)$ holds, then it is easy to see that $(\varphi \,\mathsf{U}_\mathsf{w}\, \perp) \equiv (\perp \mathsf{R}_\mathsf{w}\, \varphi) \equiv \varphi$ holds (if the automaton built from φ is a single-state loop automaton, then the automaton built from this automaton for either of these formulas using the new translation rules is structurally identical to the automaton for φ). As a nontrivial special case, however, the subclass $LTL^{\text{1-state}}(AP)$ contains all formulas of the form $\bigwedge_{1\leq i\leq n} \mathsf{GF}\theta_n \equiv \bigwedge_{1\leq i\leq n} \big(\bot \mathsf{R_w} (\top \mathsf{U_s} \theta_n)\big)$ (where $0 \leq n < \omega$, and $\theta_i \in PL(AP)$ for all $1 \leq i \leq n$).[1]

The following lemma states a further consequence of the new translation rules.

Lemma 5.6.5 *Let $\varphi_1, \varphi_2 \in LTL(AP)$ be LTL formulas which can be translated into single-state automata, and let $\theta \in PL(AP)$ be a Boolean formula. The LTL formulas $(\varphi_1 \wedge \varphi_2)$ and $(\theta \mathsf{R} \varphi_1)$ can be translated into self-loop automata with no transitions having two or more target states.*

Proof: Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_\theta$ be single-state automata built for the formulas φ_1, φ_2 and θ, respectively. Because $\mathcal{A}_1$ and $\mathcal{A}_2$ are single-state automata, no transition in either automaton has two or more target states.

The new translation rule for the $\wedge$ connective merges pairs of initial transitions of $\mathcal{A}_1$ and $\mathcal{A}_2$ into initial transitions of a compound automaton for the formula $(\varphi_1 \wedge \varphi_2)$. Let t be one of these transitions. Obviously, the transition t has at most one target state if it was built from a pair of transitions, one (or both) of which had an empty set of target states. Otherwise t was built from a pair of self-loops of the automata $\mathcal{A}_1$ and $\mathcal{A}_2$, in which case t is a self-loop of the compound automaton. Because $\mathcal{A}_1$ and $\mathcal{A}_2$ are single-state automata, it is easy to see from the definition of the new $\wedge$ rule that t has exactly one target state also in this case.

An automaton for the formula $(\theta \mathsf{R} \varphi_1)$ can be built using the universally applicable version of the optimized rule for the Release connective. Every initial transition of this automaton is either a self-loop obtained from an initial transition of $\mathcal{A}_1$ by removing the initial state of $\mathcal{A}_1$ from the target states of this transition, or a transition whose target states comprise the union of the target states of an initial transition of $\mathcal{A}_\theta$ and an initial transition of $\mathcal{A}_1$. Because $\mathcal{A}_1$ is a single-state automaton and the target state set of every transition of $\mathcal{A}_\theta$ is empty, it follows that all initial transitions of the compound automaton have at most one target state (which is either the initial state of the compound automaton, or the initial state of $\mathcal{A}_1$). $\square$

The Subclass $LTL^{\text{CND}+}$

Combining the above definition of $LTL^{\text{1-state}}(AP)$ with Lemma 4.6.2 and Lemma 5.6.5, we obtain the following syntactic definition of a subclass of formulas which can be translated (effectively using a translation procedure extended with the new translation rules for the $\wedge$, U and R connectives) into self-loop alternating automata with no transitions having two or more target states. (The proof that this is indeed the case proceeds similarly to the proof

[1] Formulas of the form $\bigwedge_{1\leq i\leq n} \mathsf{GF}\theta_n$ are sometimes used as simple *fairness constraints* to restrict verification of an LTL formula φ to computations that satisfy a set of propositional constraints infinitely often, using formulas of the form $\neg\big((\bigwedge_{1\leq i\leq n} \mathsf{GF}\theta_n) \rightarrow \varphi\big) \equiv \big((\bigwedge_{1\leq i\leq n} \mathsf{GF}\theta_n) \wedge \neg\varphi\big)$ in verification. Because any fairness constraint of this form can be translated into a single-state automaton (as observed already by Couvreur [1999]), the alternating automaton built for the formula $[\neg\varphi]^{\text{PNF}}$ combined with the fairness constraint is never more than two states larger than the automaton built for the formula $[\neg\varphi]^{\text{PNF}}$.

of Proposition 4.6.3, using Lemma 5.6.5 as an additional base case for induction.) Formally, we define the subclasss $LTL^{\text{CND}^+}(AP)$ as the smallest set of LTL formulas closed under finite application of the syntactic rule

$$\begin{gathered}\text{If } \theta \in PL(AP),\, \varphi_1, \varphi_2 \in LTL^{\text{1-state}}(AP),\text{ and } \psi_1, \psi_2 \in LTL^{\text{CND}^+}(AP),\\ \text{then}\\ \varphi_1,\, (\varphi_1 \wedge \varphi_2),\, (\theta \,\mathsf{R}\, \varphi_1),\, \mathsf{X}\psi_1,\, (\psi_1 \vee \psi_2),\, (\psi_1 \wedge \theta),\, (\theta \wedge \psi_1),\, (\theta \,\mathsf{U}\, \psi_1),\\ (\psi_1 \,\mathsf{R}\, \theta) \in LTL^{\text{CND}^+}(AP)\end{gathered}$$

(where U and R can be either weak or strong binary temporal connectives). Similarly to the class $LTL^{\text{CND}}(AP)$, $LTL^{\text{CND}^+}(AP)$ is closed under rewriting formulas into positive normal form. For example, it is straightforward to check from the syntactic definition that the formula

$$\Big[\Big(\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big) \wedge \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big)\Big) \vee \big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)\Big]$$

considered in Ex. 3.1.1 and Ex. 5.6.1 belongs to the class $LTL^{\text{CND}^+}(AP)$, and the same holds for the formula $\big(((\top \,\mathsf{U_s}\, p_1) \wedge (\top \,\mathsf{U_s}\, p_2)) \vee p_3\big)$ considered in Ex. 5.6.2.

Expressiveness and Satisfiability

Because $PL(AP) \subseteq LTL^{\text{1-state}}(AP)$ holds, it is easy to see from the syntactic definition that $LTL^{\text{CND}}(AP) \subseteq LTL^{\text{CND}^+}(AP)$ holds. Therefore, every LTL formula expressible in $LTL^{\text{CND}}(AP)$ is trivially logically equivalent to a formula in $LTL^{\text{CND}^+}(AP)$. This property does not hold in the converse direction, however: the class $LTL^{\text{CND}^+}(AP)$ is strictly more expressive than the subclass $LTL^{\text{CND}}(AP)$ for any nonempty set of atomic propositions AP. (We prove this result only indirectly, referring to known results [Clarke and Draghicescu 1989; Maidl 2000a] on the relative expressive power of LTL and the branching time temporal logics CTL [Clarke and Emerson 1982a,b] and CTL* [Emerson and Halpern 1983, 1986]; see, for example, [Emerson 1990] for an introduction into the syntax and semantics of these branching time logics.)

Proposition 5.6.6 *Let AP be a nonempty set of atomic propositions. The subclass $LTL^{\text{CND}^+}(AP)$ is strictly more expressive than $LTL^{\text{CND}}(AP)$.*

Proof: Let $p \in AP$ be an atomic proposition. Consider the LTL formula $\mathsf{GF}\neg p \equiv \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, \neg p)\big) \in LTL(AP)$. Obviously, because $(\top \,\mathsf{U_s}\, \neg p) \in LTL^{\text{1-state}}(AP)$ holds, $\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, \neg p)\big) \in LTL^{\text{CND}^+}(AP)$ holds by the syntactic definition of $LTL^{\text{CND}^+}(AP)$. We show that no formula in the subclass $LTL^{\text{CND}}(AP)$ is logically equivalent to this formula.

Suppose that there exists a formula $\varphi \in LTL^{\text{CND}}(AP)$ such that $\varphi \equiv \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, \neg p)\big) \equiv \mathsf{GF}\neg p$ holds. By Proposition 4.6.6, φ is also logically equivalent to the formula $[\varphi]^{\overline{\text{det}}} \in LTL^{\overline{\text{det}}}(AP)$ obtained from φ by applying the rewrite rules presented in Sect. 4.6.4. Let $\psi \overset{def}{=} \big[\neg[\varphi]^{\overline{\text{det}}}\big]^{\text{PNF}}$ be the positive normal form of $\neg[\varphi]^{\overline{\text{det}}}$; clearly, $\psi \equiv \neg\varphi \equiv \neg\mathsf{GF}\neg p \equiv \mathsf{FG}p$ holds, and by the definition of $LTL^{\overline{\text{det}}}(AP)$, ψ belongs to the subclass $LTL^{\text{det}}(AP)$

of Maidl [2000a]. By Corollary 1 of [Maidl 2000a], there exists a formula ψ' in the branching time logic CTL such that the model checking problems for the formula ψ' and the CTL* formula $\mathsf{A}\psi \equiv \mathsf{A}(\mathsf{FG}p)$ have the same answer in every state in every structure. This is, however, a contradiction, because the formula $\mathsf{A}(\mathsf{FG}p)$ is known to be not equivalent in this sense to any formula in the logic CTL [Clarke and Draghicescu 1989]. Therefore, our assumption that $\varphi \in LTL^{\mathrm{CND}}(AP)$ holds is incorrect, and the result follows. □

Let $\varphi \in LTL^{\mathrm{CND}^+}(AP)$ be a formula. Because the translation procedure extended with the rules presented in this chapter still maps all Boolean formulas into single-state automata, and because each application of a new translation rule adds one new state to the constructed automaton, the number of states in an automaton built from $[\varphi]^{\mathrm{PNF}}$ is bounded by the length of φ. Clearly, also the number of acceptance conditions in this automaton still remains bounded by $|\varphi|$: identically to the basic translation, the new translation rules introduce new acceptance conditions only for binary temporal subformulas with a strong main connective. By Lemma 4.6.1, the automaton can be completed into a nondeterministic automaton with at most $1 + |\varphi|$ states. Proposition 4.6.10 now applies to show that formulas in the class $LTL^{\mathrm{CND}^+}(AP)$ have the same "small model" property as the formulas in the class $LTL^{\mathrm{CND}}(AP)$. Therefore, a consideration identical to the one in the proof of Corollary 4.6.11 shows that also the decision problem for satisfiability in $LTL^{\mathrm{CND}^+}(AP)$ is $\mathbf{NP}$-complete.

Proposition 5.6.7 *Let AP be a countably infinite set of atomic propositions. The satisfiability problem for $LTL^{\mathrm{CND}^+}(AP)$ is $\mathbf{NP}$-complete.*

As noted in Sect. 4.6.5, the satisfiability of a CTL formula obtained from a formula $\varphi \in LTL^{\mathrm{CND}}(AP)$ by prefixing all of its temporal operators with the existential CTL path quantifier implies the existence of a nonbranching tree (i.e., a word) model for the CTL formula, and this model can be identified with a model for φ. This correspondence between $\exists$CTL satisfiability and the existence of word models does not directly carry over to $\exists$CTL formulas obtained from those in the subclass $LTL^{\mathrm{CND}^+}(AP)$ using the same conversion, however: for example, applying the conversion to the LTL formula

$$(\mathsf{F}p \wedge \mathsf{G}\neg p) \equiv \big((\top \,\mathsf{U_s}\, p) \wedge (\bot \,\mathsf{R_w}\, \neg p)\big) \in LTL^{\mathrm{CND}^+}\big(\{p\}\big)$$

yields the satisfiable $\exists$CTL formula $\big(\mathsf{E}(\top \,\mathsf{U_s}\, p) \wedge \mathsf{E}(\bot \,\mathsf{R_w}\, \neg p)\big)$, which nevertheless has no word model because the LTL formula is unsatisfiable. Hence, the $\mathbf{NP}$-completeness of $\exists$CTL satisfiability [Kupferman and Vardi 1995, 2000] does not directly allow to conclude the above $\mathbf{NP}$-completeness result for satisfiability in $LTL^{\mathrm{CND}^+}(AP)$ unlike in the case for $LTL^{\mathrm{CND}}(AP)$.

6 REMOVING REDUNDANT TRANSITIONS

The refined translation rules introduced in the previous chapter can be seen as heuristics for simplifying self-loop alternating automata built using the basic rules of Sect. 3.1 by replacing some of their transitions with transitions having fewer target states without changing the language of the automata. The refined rules do not explicitly aim for reducing the actual number of states or transitions in an automaton, however; all such reductions (if any) arise only as indirect side effects of applying the refined rules. In this chapter, we shall explore techniques for detecting *redundant* transitions that can be removed from a self-loop alternating automaton without changing its language. Formally, a transition $t \in \Delta$ of an automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ is *(fin-)redundant* iff $\mathcal{A}$ is fin-equivalent to the automaton $\mathcal{A}' = \big\langle \Sigma, Q, \Delta \setminus \{t\}, q_I, \mathcal{F} \big\rangle$ obtained from $\mathcal{A}$ by removing the transition t from Δ. (Clearly, if $\mathcal{A}$ is a self-loop alternating automaton, then so is $\mathcal{A}'$, because the automaton $\mathcal{A}'$ obviously has at most as many simple cycles as $\mathcal{A}$. Therefore, all heuristics specific to the simplification of self-loop alternating automata remain applicable to $\mathcal{A}'$ and any automaton obtained from it by removing more transitions.)

The incremental translation procedure for building increasingly complex automata from simpler automata can easily be combined with on-the-fly transition redundancy analysis [Gastin and Oddoux 2001]. Because the structure of every automaton $\mathcal{A}$ built using a rule in the translation procedure from LTL into automata remains unchanged in any further application of a rule that uses $\mathcal{A}$ as a component, the language of $\mathcal{A}$ remains fixed regardless of the way the translation rules are used after $\mathcal{A}$ has been defined. Therefore, the automaton $\mathcal{A}$ can be scanned for redundant transitions immediately after constructing it. In particular, removing all redundant initial transitions of $\mathcal{A}$ before applying another translation rule reduces the effort needed for building a compound automaton in which $\mathcal{A}$ occurs as a component, as well as any automaton built incrementally from these component automata. This kind of on-the-fly transition redundancy analysis is conceptually more difficult to combine with tableau-based procedures for translating LTL directly into nondeterministic automata due to the "top-down" approach used in these procedures to construct the automaton. The requirement for access to a complete automaton for φ is implicit also in the design of many minimization techniques based on the use of simulation relations [Etessami and Holzmann 2000; Somenzi and Bloem 2000; Etessami et al. 2001, 2005; Etessami 2002; Fritz and Wilke 2002, 2005; Gurumurthy et al. 2002].

We shall concentrate on the detection of redundant initial transitions of self-loop alternating automata due to its above potential benefits in reducing the effort needed for the incremental application of the translation rules. Obviously, this restriction will in the general case miss some opportunities for removing transitions: even though the translation procedure is "modular" in the syntactic structure of the formula, it is not modular in the sense that every nonredundant initial transition of an automaton would remain nonredundant in automata built from it incrementally. For example, the automaton built from the formula $\big(\mathsf{F}p_1 \vee \mathsf{X}(\mathsf{G}p_1 \vee p_2)\big) \in LTL\big(\{p_1, p_2\}\big)$ with the basic

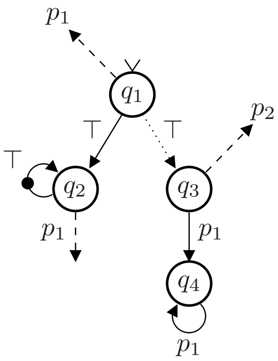


Fig. 6.1: A self-loop alternating automaton $\mathcal{A}$ built for the formula $\big(\mathsf{F}p_1 \vee \mathsf{X}(\mathsf{G}p_1 \vee p_2)\big)$. The transition from the state q_3 to the state q_4 is redundant in $\mathcal{A}$ even though it is not redundant in $\mathcal{A}^{q_3}$

translation rules has redundant (non-initial) transitions even though no subautomaton rooted at one of its non-initial states has any redundant transitions (see Fig. 6.1). Of course, this phenomenon is hardly surprising because of the obvious analogy between incremental transition redundancy analysis and the task of simplifying a compound LTL formula built from one or two arbitrary subformulas and a connective. Nevertheless, the "local" approach to the detection of redundant transitions has, besides special cases that are easier to check than the general case, advantages that ease the implementation of the translation procedure. For example, restricting redundancy analysis to the transitions starting from the initial state of an automaton will trivially preserve the correspondence between the states of the automaton and the node subformulas of the formula under translation. Therefore the subautomata built for these subformulas remain directly accessible in case they are needed again in the translation if some of these subformulas occur multiple times in the formula.

6.1 REDUNDANT TRANSITIONS AND LANGUAGE CONTAINMENT

It is clear that any transformation used for simplifying an alternating automaton $\mathcal{A}$ is correct (in the sense that it preserves the language fin-recognized by the automaton) iff $\mathcal{A}$ and the automaton $\mathcal{A}'$ built in the transformation satisfy the language containment relationships

$$\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}') \qquad \text{and} \qquad \mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}).$$

In particular, if the automaton $\mathcal{A}'$ is obtained from $\mathcal{A}$ by removing transitions, it is easy to see that the right-hand condition holds trivially between the automata, since all fin-accepting runs of $\mathcal{A}'$ are fin-accepting runs of $\mathcal{A}$ in this case. In transition redundancy analysis, it is thus sufficient to check that the left-hand language containment relationship holds between the automata. By the classic reformulation of language containment (see Sect. 5.2), this test involves finding an automaton for the language $\overline{\mathcal{L}_{\text{fin}}(\mathcal{A}')}$. In the language containment based optimizations of Sect. 5.3 and Sect. 5.5, a self-loop alternating automaton that recognizes the complement of the language of another self-loop alternating automaton could be found by reusing the translation procedure from LTL into automata because the automaton had been originally built for a known (subformula of an) LTL formula. However, the present language containment problem differs from these previous cases in that the automaton $\mathcal{A}'$ is now obtained directly from another automaton instead of an LTL formula. Even if the automaton $\mathcal{A}$ did correspond to a known LTL formula, removing a transition from it may break this correspondence, and checking whether this is the case is merely a restatement of the language containment problem.

If the given automaton $\mathcal{A}$ is a self-loop alternating automaton, then obviously also the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by removing one of its transitions is such an automaton. Therefore, an LTL formula ψ corresponding to the automaton $\mathcal{A}'$ can be found, for example, via the reverse translation discussed in Sect. 3.4. In principle, an automaton for the complement language $\overline{\mathcal{L}_{\mathrm{fin}}(\mathcal{A}')}$ can be then obtained as before by applying the basic translation procedure to the positive normal form of $\neg\psi$. Unfortunately, as noted in the discussion at the end of Sect. 3.4, a reverse translation procedure based on the incremental application of the pattern given in Lemma 3.4.1 may yield an LTL formula with exponentially many syntactically distinct subformulas in the number of states in the automaton $\mathcal{A}'$, and thus translating the negation of this formula back into an automaton using the basic translation procedure requires exponential time. It is nevertheless not always necessary to apply the reverse translation to all states of the automaton $\mathcal{A}'$: because $\mathcal{A}$ and $\mathcal{A}'$ are self-loop alternating automata that differ only in the transitions leaving the source state q of the transition that was removed from $\mathcal{A}$, the automaton $\mathcal{A}'$ shares with the automaton $\mathcal{A}$ all subautomata that do not include the state q. If the automaton $\mathcal{A}$ was built from an LTL formula φ using the translation rules, it may therefore be possible to substitute some formulas in a reverse translation pattern directly with subformulas of φ corresponding to these subautomata. It is thus sufficient to apply the reverse translation to the state q in addition to all states of which q is a descendant in $\mathcal{A}'$; in other words, the cost of reverse translation increases with the length of the longest path from the initial state of $\mathcal{A}'$ to the state q. If the redundancy analysis is restricted to the initial transitions of $\mathcal{A}$, however, the formula ψ can be found from $\mathcal{A}'$ in a single reverse translation step applied to the initial state of the automaton.

Even though some subformulas of φ can be reused in the reverse translation procedure for finding an LTL formula ψ corresponding to the automaton $\mathcal{A}'$, the automata built for these subformulas cannot in the general case be reused in the translation of the positive normal form of $\neg\psi$ back into an automaton without repeating the translation procedure. Although this requirement is in fact common to all language containment checks presented, the reverse translation used in the present case is nevertheless likely to introduce formulas that are not subformulas of φ nor the positive normal form of $\neg\varphi$. Because the formula ψ also depends on the particular transition chosen for the redundancy analysis, it becomes difficult to estimate beforehand the actual number of formula translation subproblems that will arise during the construction and simplification of a self-loop alternating automaton built from the formula φ. In the next section, we examine special cases of transition redundancy analysis in which the reverse translation can be avoided by dividing the language containment test $\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}')$ into several subproblems. These special cases allow for limited transition redundancy analysis while keeping the formula translation subproblems within the set of node subformulas of φ and (the positive normal forms of) their negations.

6.2 DETECTING REDUNDANT INITIAL TRANSITIONS BY TRANSITION SUBSTITUTION

In this section, we develop heuristics for detecting redundant initial transitions in self-loop alternating automata without solving the general language containment problem presented in Sect. 6.1. Our results combine local structural analysis of the automata with language containment checks that can be handled by applying the basic translation only to node subformulas of a given LTL formula or their negations. In some cases, however, we have to trade the language containment test between two languages (as described in the previous section) for a more general one that involves containment between set intersections of languages. We present our results using arbitrary self-loop alternating automata; their application to self-loop alternating automata with acceptance synchronized runs will be discussed in Sect. 6.2.5.

6.2.1 Redundant Transitions and Runs of an Automaton

Rephrasing the criterion on transition redundancy as a condition on runs of an automaton $\mathcal{A}$, we see a transition to be redundant iff, for every accepting run of $\mathcal{A}$ that contains an edge labeled with the transition, there exists another accepting run (on the same input) in which the automaton avoids taking this transition. This high-level intuition of finding accepting runs in which the automaton avoids taking a given transition forms the basic strategy of proving the results of this section by modifying accepting runs of $\mathcal{A}$ that contain edges labeled with the transition into accepting runs that do not contain such edges.

By the above characterization, a transition is obviously redundant if it never occurs in any accepting run of the automaton. We have already used this fact previously in Corollary 2.3.11 and Sect. 5.1.2 for removing transitions with an empty guard (characterizable by an unsatisfiable Boolean formula in $PL(AP)$ in automata having the alphabet 2^{AP}), or transitions that spawn a collection of subautomata, the intersection of whose languages is empty.

Obviously, all redundant transitions of self-loop alternating automata do not necessarily fall into the above category, and the automaton may well have accepting runs, some edges of which are labeled with redundant transitions. In the following, we shall investigate conditions under which these runs can be modified into accepting runs in which the automaton avoids taking such transitions. Formally, we shall often make use of the following result that characterizes an obvious way to extract a semi-run avoiding a given transition from any run of the automaton by truncating every branch of the run at the first occurrence of an edge labeled with the transition. If this semi-run can then be extended back into an accepting run (on the same input) that still avoids the transition, it follows that the transition is redundant if the same extension result holds for all words in the language of the automaton.

Lemma 6.2.1 *Let $G = \langle V, E, L\rangle$ be a fin-accepting run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ on some $w \in \Sigma^\omega$, and let $t \in \Delta$ be a transition of $\mathcal{A}$ with source state $q \in Q$. The graph $G' = \langle V', E', L'\rangle$, where*

- $V'_0 \overset{def}{=} \{v_0\}$, $V'_{i+1} \overset{def}{=} \bigcup_{v \in V'_i} \{V'' \subseteq V_{i+1} \,|\, \langle v, V''\rangle \in E, L(\langle v, V''\rangle) \neq t\}$ *for all* $0 \leq i < \omega$,
- $E' \overset{def}{=} \{\langle v, V''\rangle \in E \,|\, \{v\} \cup V'' \subseteq V', L(\langle v, V''\rangle) \neq t\}$, *and*
- $L'(x) \overset{def}{=} L(x)$ *for all* $x \in V' \cup E'$,

is a fin-accepting semi-run of $\mathcal{A}$ on w such that none of the edges of E' is labeled with the transition t, and every node of G' with no outgoing edges is labeled with the state q.

Proof: Obviously, $V' \subseteq V$ (with $V'_i \subseteq V_i$ for all $0 \leq i < \omega$) and $E' \subseteq E$ hold; because G is a run, it follows that G' can be partitioned into finite disjoint levels with edges between successive levels of G'.

Because G is a run of $\mathcal{A}$, $L'(v_0) = L(v_0) = q_I$ holds. Let $v \in V'$. Because G is a run and $V' \subseteq V$ holds, v has a unique consistently labeled outgoing edge $e \in E$ in G. Because $E' \subseteq E$ holds, v now has either no outgoing edges in G', or v keeps its unique outgoing edge; by the definition of E', this edge is always labeled with a transition different from t. Because $L'(x) = L(x)$ holds for all $x \in V' \cup E'$, the labeling of e is still consistent in G'.

Let $v' \in V'_i$ for some $1 \leq i < \omega$. From the definition of V' it follows that there exists a node $v \in V'_{i-1}$ and an edge $e = \langle v, V''\rangle \in E$, $v' \in V''$, such that $L(e) \neq t$ holds. Therefore $e \in E'$ holds, and G' is a semi-run of $\mathcal{A}$ on w.

Clearly, because every infinite branch $\beta' \in \mathcal{B}(G')$ is also an infinite branch in G and $L'(e) = L(e)$ holds for all $e \in E'$, $\text{fin}(\beta') = \emptyset$ holds in both G and G', and thus G' is a fin-accepting semi-run of $\mathcal{A}$.

Finally, if $v \in V'$ has no outgoing edges, then the unique edge starting from v in G is labeled with the transition t, and because the labeling of G is consistent, v is labeled with the source state of t in G, i.e., $L(v) = q = L'(v)$ holds. $\square$

6.2.2 Transition Substitution

Obviously, a run of a self-loop alternating automaton can be modified to avoid a given transition, for example, if the automaton has another transition that can be substituted for every occurrence of the given transition in the run. Clearly, such a substitution can be made only if the transitions share their source state, and if the automaton could in fact have taken either of the transitions at each occurrence of the given transition in the run. Formally, given two transitions $t = \langle q, \Gamma, F, Q'\rangle \in \Delta$ and $t' = \langle q', \Gamma', F', Q''\rangle \in \Delta$ of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ and a set $\widehat{\Gamma} \subseteq \Gamma$ of symbols from the automaton's alphabet, we say that the transition t' is $\widehat{\Gamma}$-*substitutable* *for* t (*in a run of* $\mathcal{A}$) iff $t \neq t'$, $q = q'$ and $\widehat{\Gamma} \subseteq \Gamma'$ hold. The transitions t and t' are called the *substituted* and the *substituting* transitions, respectively. (If the guards $\widehat{\Gamma}$ and Γ' are represented in an automaton with the alphabet 2^{AP} using their characteristic Boolean formulas $\hat{\theta} \in PL(AP)$ and $\theta' \in PL(AP)$, respectively, then $\widehat{\Gamma} \subseteq \Gamma'$ holds iff the propositional implication $(\hat{\theta} \rightarrow \theta')$ is valid.)

Let t and t' be two transitions of an alternating automaton $\mathcal{A}$ with the components specified above. Obviously, renaming an edge labeled with the transition t in a run of $\mathcal{A}$ with the transition t' (using the above criterion for substitutability) necessitates that t and t' share their set of target states to keep the labeling of the run consistent. In accepting runs of the automaton, however, a less restrictive strategy for substituting the transition t' for an occurrence of the transition t is to require (besides substitutability) that the subautomata rooted at the target states of t' accept the remainder of the input beginning at the level following the occurrence of t. Intuitively, the run could thus be first modified into an accepting semi-run of $\mathcal{A}$ (applying the principle in Lemma 6.2.1 to remove the occurrence of t from the run), and then extended back into an accepting run of $\mathcal{A}$ using accepting runs of subautomata rooted at the target states of t' on the rest of the input (Proposition 2.3.14). Formally, for all $\widehat{\Gamma} \subseteq \Gamma$, we say that the transition t' is $\widehat{\Gamma}$-substitutable for t *under fin-acceptance* iff t' is $\widehat{\Gamma}$-substitutable for t', and $\bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \subseteq \bigcap_{q' \in Q''} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ holds.

6.2.3 Substitutability and Redundant Initial Transitions of Self-loop Automata

Substitutability of transitions under fin-acceptance suggests a general strategy for detecting redundant transitions in an alternating automaton by finding sufficient conditions under which every occurrence of an edge labeled with a transition $t = \langle q, \Gamma, F, Q' \rangle$ in a fin-accepting run of the automaton can be replaced (along with the fragment of the run starting from the source node of the edge) for all $\sigma \in \Gamma$ with an edge labeled with another transition that is $\{\sigma\}$-substitutable for the transition t under fin-acceptance. Unfortunately, the local criteria on substitutability under fin-acceptance are not always sufficient to guarantee this global substitutability property if some of the languages recognized by the subautomata rooted at the target states of a substituting transition depend on the substituted transition. For example, even though the $\bullet$-transition of the automaton shown on the left in Fig. 5.8 (c) (p. 146) is substitutable for the transition with no acceptance conditions under fin-acceptance for all models of the guard p_1, the run obtained from a fin-accepting run of the automaton by substituting the $\bullet$-transition for every occurrence of the other transition is no longer fin-accepting.

Because the language of an automaton depends only on the subautomaton rooted at its initial state (Proposition 2.3.12), checking that the source state of a substituted transition is neither a target state of a substituting transition (under fin-acceptance) nor reachable from any of these states in the automaton provides a sufficient additional condition to ensure that the language of the automaton is not changed by the removal of the substituted transition. In self-loop alternating automata, this property follows immediately if, for all symbols σ in the guard of the substituted transition, there exists a non-self-loop transition that is $\{\sigma\}$-substitutable for it under fin-acceptance.

Proposition 6.2.2 *Let $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ be a transition of a self-loop alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$. Assume that there exists a nonempty set of non-self-loop transitions $T = \{t_1, t_2, \ldots, t_n\} \subseteq \Delta \setminus \{t\}$ (for some $1 \leq n < \omega$ such that $t_i = \langle q, \Gamma_i, F_i, Q'_i \rangle$ and $q \notin Q'_i$ hold for all $1 \leq i \leq n$), such that for all $\sigma \in \Gamma$, there exists a transition $t' \in T$ that is $\{\sigma\}$-substitutable for t under fin-acceptance. The automaton $\mathcal{A}$ is fin-equivalent to the automaton $\mathcal{A}' \stackrel{\text{def}}{=} \langle \Sigma, Q, \Delta \setminus \{t\}, q_I, \mathcal{F} \rangle$ obtained from $\mathcal{A}$ by removing the transition t from Δ.*

Proof: As noted in Sect. 6.1, $\mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds trivially. To show language containment in the other direction, let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$. By Lemma 6.2.1, we can extract from a fin-accepting run $G = \langle V, E, L \rangle$ of $\mathcal{A}$ on w a fin-accepting semi-run $G' = \langle V', E', L' \rangle$ that contains no edges labeled with the transition t. Obviously, G' is then also a fin-accepting semi-run of $\mathcal{A}'$ on w.

Let $v \in V'_i$ for some $0 \leq i < \omega$ be a node with no outgoing edges in G'; by the definition of G', the unique outgoing edge of v in G is labeled with the transition $t = \langle q, \Gamma, F, Q' \rangle$. Because L is consistent, $w(i) \in \Gamma$ holds, and the union of the labels of the successors of v in G is equal to Q', and because G is fin-accepting, it follows (Proposition 2.3.9) that $\mathcal{A}^{q'}$ fin-accepts w^{i+1} for all $q' \in Q'$, i.e., $w^{i+1} \in \bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ holds.

Because $w(i) \in \Gamma$ holds, there exists an index $1 \leq j \leq n$ such that the transition $t_j = \langle q, \Gamma_j, F_j, Q'_j \rangle \in T$ is $\{w(i)\}$-substitutable for t under fin-acceptance. Therefore $w(i) \in \Gamma_j$ and $w^{i+1} \in \bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \subseteq \bigcap_{q' \in Q'_j} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ hold, and it follows that $\mathcal{A}^{q'}$ fin-accepts w^{i+1} also for all $q' \in Q'_j$. By Proposition 2.3.15, it follows that the automaton $\mathcal{A}^q$ has a fin-accepting run on w^i, where the edge starting from the node at level 0 of this run is labeled with the transition t_j, and the target nodes of this edge are labeled with the states in Q'_j. By Proposition 2.3.6, all states in this run (except possibly the node at level 0) are labeled with descendants of the state q. But then, because $q \notin Q'_j$ holds and $\mathcal{A}^q$ is a self-loop alternating automaton, it follows that no state at a level greater than 0 is labeled in this run with the state q, either, and because $t_j \neq t$ holds, no edge in this run is labeled with the transition t. It follows that the run is a fin-accepting run of $(\mathcal{A}')^q$ on w^i.

Because the above result holds for all nodes of G' with no outgoing edges, the fin-accepting semi-run G' can be extended into a fin-accepting run of $\mathcal{A}'$ on w by Proposition 2.3.14. Therefore $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}')$ holds, and it follows that $\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}')$ holds. □

If a substituting transition is a self-loop, it includes its own source state in its target states. Because also the substituted transition is rooted at this state, it may not be safe to remove the substituted transition from the automaton without changing the language of the subautomaton rooted at the common source state of the transitions. It is in some cases nevertheless possible to find the substituted transition to be redundant without resorting to, for example, reverse translation based transition redundancy analysis.

Consider an infinite branch in a fin-accepting run of a self-loop alternating automaton. If the source state of a substituted self-loop is a transient state of the automaton, then every self-loop starting from this state can occur as the label of only finitely many edges in the branch (cf. Corollary 2.3.19). Consequently, no substituting transition (under fin-acceptance) can contribute to the acceptance conditions occurring infinitely many times along this branch, either, regardless of whether the transition is a self-loop or not. The requirements on transition substitution under fin-acceptance now guarantee that the language of the subautomaton rooted at the transient state is preserved if the substituted self-loop is removed from the automaton.

In case the source state of the substituted self-loop is not a transient state in the automaton, the conditions on substitutability under fin-acceptance are not by themselves sufficient to allow the self-loop to be safely removed from the automaton such that the language of the automaton is preserved. Because transition substitution does not depend on the acceptance conditions of the transitions, it may occur, for example, that the nontransient source state of the substituted self-loop becomes transient when this self-loop is removed from the automaton. Consequently, the language of the automaton may change in the modification, for example, if the new transient state has no other successors than itself. Obviously, this problem will not arise if the set of acceptance conditions of the substituting transition is a subset of the conditions of the substituted transition. This requirement can be generalized slightly: the substituted self-loop can safely be removed if there exists (for every symbol σ in the guard of the substituted transition) a set of transitions, all of which are $\{\sigma\}$-substitutable for the transition under fin-acceptance such that the transitions do not share any acceptance conditions that are not included also in the acceptance conditions of the substituted self-loop. (In runs of self-loop alternating automata, the acceptance conditions of the substituting transitions can affect fin-acceptance only in those infinite branches that converge to the common source state of the transitions. Therefore, for example, if an infinite branch of an accepting run ends in an infinite suffix in which the automaton simply repeats the substituted transition, then every occurrence of this transition can be replaced with a finite sequence of substituting transitions. The properties of the sets of substituting transitions guarantee that the substitution can be done throughout the entire suffix without violating fin-acceptance.)

The above informal discussion can be summarized as the following result.

Proposition 6.2.3 *Let $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ be a self-loop transition of a self-loop alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$. Assume that, for all $\sigma \in \Gamma$, there exists a nonempty set of transitions $T_\sigma = \{t_{\sigma,1}, t_{\sigma,2}, \ldots, t_{\sigma,n_\sigma}\} \subseteq \Delta \setminus \{t\}$ (for some $1 \leq n_\sigma < \omega$) with acceptance conditions $F_{\sigma,1}, F_{\sigma,2}, \ldots, F_{\sigma,n_\sigma} \subseteq \mathcal{F}$ (respectively) such that $t_{\sigma,i}$ is $\{\sigma\}$-substitutable for t under fin-acceptance for all $1 \leq i \leq n_\sigma$, and either q is a transient state of $\mathcal{A}$, or $\bigcap_{1 \leq i \leq n_\sigma} F_{\sigma,i} \subseteq F$. The automaton $\mathcal{A}$ is fin-equivalent to the automaton $\mathcal{A}' \stackrel{def}{=} \big\langle \Sigma, Q, \Delta \setminus \{t\}, q_I, \mathcal{F} \big\rangle$ obtained from $\mathcal{A}$ by removing the transition t from Δ.*

<u>*Proof:*</u> As noted in Sect. 6.1, $\mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A})$ holds trivially. We show that the language inclusion holds also in the converse direction. Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$, and let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}$ on w. By Lemma 6.2.1, the run G can be modified into a fin-accepting semi-run $G' = \langle V', E', L' \rangle$

that does not contain edges labeled with the transition t; obviously, G' is then a fin-accepting semi-run of the automaton $\mathcal{A}'$ on w. Our goal is to show that G' can be extended into a fin-accepting run of $\mathcal{A}'$ on w. Because all nodes of G' with no outgoing edges are labeled with the state q (by the construction in Lemma 6.2.1), it needs to be shown that the automaton $(\mathcal{A}')^q$ fin-accepts w^i for all $0 \leq i < \omega$ for which V_i' contains a node with no outgoing edges in G': the result then follows directly by applying Proposition 2.3.14.

(Construction of a fin-accepting run of $(\mathcal{A}')^q$) Let $v \in V_i' \subseteq V_i$ be a node with no outgoing edges in G' for some $0 \leq i < \omega$. Because $L'(v) = L(v) = q$ holds, v is (in G) the initial node of a fin-accepting run of $\mathcal{A}^q$ on w^i embedded in G (Proposition 2.3.9). By Proposition 2.3.7, there exists an index $0 \leq j \leq \omega$ such that this run contains a chain of edges $(e_k)_{0 \leq k < j+1}$, $e_k \in E \cap (V_{i+k} \times 2^{V_{i+k+1}})$, where the transition $L(e_k) = \langle q, \Gamma_k, F_k, Q_k' \rangle \in \Delta$ is an initial self-loop of $\mathcal{A}^q$ for all $0 \leq k < j$, and if $j < \omega$ holds, then the transition $L(e_j) = \langle q, \Gamma_j, F_j, Q_j' \rangle \in \Delta$ is an initial transition of $\mathcal{A}^q$ that is not a self-loop. We shall use this chain of edges to define a chain of initial transitions of $\mathcal{A}^q$ (not including the transition t) that satisfies the fin-acceptance condition. This chain of transitions will then be used to define a fin-accepting run of $(\mathcal{A}')^q$ on w. To ensure that the fin-acceptance condition is always satisfied, we need to treat every acceptance condition not included in t's acceptance conditions F fairly in the construction. For this purpose, write $\mathcal{F} \setminus F = \{f_1, f_2, \ldots, f_m\}$ for some $0 \leq m < \omega$; in addition to the sequence of transitions, we define a sequence of integers $c_0, c_1, c_2, \ldots$ to guide the definition of the chain of transitions. Let $c_0 \stackrel{def}{=} 1$.

(Construction of a chain of transitions) Assume that c_k has been defined for some $0 \leq k < j + 1$, and $1 \leq c_k \leq \max\{1, m\}$ holds. (This is obviously the case if $k = 0$.) If $L(e_k) \neq t$ holds, let $t_k \stackrel{def}{=} L(e_k)$ and $c_{k+1} \stackrel{def}{=} c_k$. Otherwise, if $L(e_k) = t = \langle q, \Gamma, F, Q' \rangle$ holds, write $w^i(k) = \sigma_k$; because the labeling L is consistent, $\sigma_k \in \Gamma$ holds. By the assumption, there exists an integer $1 \leq n_{\sigma_k} < \omega$ and a nonempty set of transitions $T_{\sigma_k} = \{t_{\sigma_k,1}, t_{\sigma_k,2}, \ldots, t_{\sigma_k,n_{\sigma_k}}\} \subseteq \Delta \setminus \{t\}$ (with acceptance conditions $F_{\sigma_k,1}, F_{\sigma_k,2}, \ldots, F_{\sigma_k,n_{\sigma_k}} \subseteq \mathcal{F}$, respectively) such that every transition in T_{σ_k} is $\{\sigma_k\}$-substitutable for t under fin-acceptance. In this case we choose the transition t_k from the set T_{σ_k} as follows:

- If q is a transient state of $\mathcal{A}$ or $m = 0$, let $t_k \in T_{\sigma_k}$ be any transition in T_{σ_k}, and define $c_{k+1} \stackrel{def}{=} c_k$.
- Otherwise, if q is not a transient state of $\mathcal{A}$ and $m \geq 1$ holds, then $\bigcap_{1 \leq \ell \leq n_{\sigma_k}} F_{\sigma_k,\ell} \subseteq F$ holds by the assumption. It is easy to see that there exists, for all $f \in \{f_1, f_2, \ldots, f_m\}$, a transition $t_{\sigma_k}^f \in T_{\sigma_k}$ that is not an f-transition of $\mathcal{A}$. In particular, because $1 \leq c_k \leq m$ holds, the transition $t_{\sigma_k}^{f_{c_k}} \in T_{\sigma_k}$ is not an f_{c_k}-transition. In this case we define $t_k \stackrel{def}{=} t_{\sigma_k}^{f_{c_k}}$ and $c_{k+1} \stackrel{def}{=} (c_k \bmod m) + 1$.

It is easy to see that $1 \leq c_{k+1} \leq \max\{1, m\}$ holds by the construction. If the transition t_k is not a self-loop, we stop the construction; otherwise we repeat the same steps to define the transition t_{k+1}. Obviously, the inductive construction always ends after a finite number of steps if $j < \omega$ holds (in this case, the transition $L(e_j) \neq t$ is not an initial self-loop of $\mathcal{A}^q$, so the construction ends at the latest after the transition t_j has been defined.) In summary, we thus find an index $0 \leq \ell \leq j$ such that the transition t_k is an initial self-loop of $\mathcal{A}^q$ for all $0 \leq k < \ell$, and if $\ell < \omega$, then t_ℓ is an initial transition of $\mathcal{A}^q$ which is not a self-loop. Furthermore, it is easy to see that $t_k \neq t$ holds for all $0 \leq k < \ell + 1$ by the construction.

(Definition of a semi-run $\widehat{G}$ of $(\mathcal{A}')^q$ on w^i) Write $t_k = \langle q, \widehat{\Gamma}_k, \widehat{F}_k, \widehat{Q}'_k\rangle$ and $\widehat{Q}'_k \setminus \{q\} = \{\hat{q}_{k,1}, \ldots, \hat{q}_{k,n_k}\}$ (for some $0 \leq n_k < \omega$) for all $0 \leq k < \ell$, and if $\ell < \omega$ holds, write $\widehat{Q}'_\ell = \{\hat{q}_{\ell,1}, \ldots, \hat{q}_{\ell,n_\ell}\}$ for some $0 \leq n_\ell < \omega$. Define the graph $\widehat{G} \stackrel{def}{=} \langle \widehat{V}, \widehat{E}, \widehat{L}\rangle$, where

- $\widehat{V}_0 \stackrel{def}{=} \{\hat{v}_0\}$, $\widehat{V}_{k+1} \stackrel{def}{=} \{\hat{v}_{k,1}, \ldots, \hat{v}_{k,n_k}\} \cup \begin{cases}\{\hat{v}_{k+1}\} & \text{if } k < \ell \\ \emptyset & \text{otherwise}\end{cases}$ for all $0 \leq k < \ell + 1$, and if $\ell < \omega$, let $\widehat{V}_k \stackrel{def}{=} \emptyset$ for all $\ell + 1 < k < \omega$;
- $\widehat{E} \stackrel{def}{=} \bigcup_{0 \leq k < \ell+1} \big\{\langle \hat{v}_k, \widehat{V}_{k+1}\rangle\big\}$; and
- for all $0 \leq k < \ell+1$, let $\widehat{L}(\hat{v}_k) \stackrel{def}{=} q$, $\widehat{L}(\hat{v}_{k,k'}) \stackrel{def}{=} \hat{q}_{k,k'}$ for all $1 \leq k' \leq n_k$, and $L\big(\langle \hat{v}_k, \widehat{V}_{k+1}\rangle\big) \stackrel{def}{=} t_k$.

($\widehat{G}$ is a semi-run of $(\mathcal{A}')^q$ on w^i)

(Partitioning) Clearly, $\widehat{V}_0 = \{\hat{v}_0\}$ is a singleton, and $\widehat{G}$ consists of finite disjoint levels with edges between successive levels of $\widehat{G}$.

(Causality) Let $\hat{v} \in \widehat{V}_k$ for some $0 \leq k < \ell + 1$. Then $\hat{v}$ either has no outgoing edges, or $\hat{v} = \hat{v}_k$ holds, and $\hat{v}$ has the unique outgoing edge $\langle \hat{v}, \widehat{V}_{k+1}\rangle \in \widehat{E}$. Furthermore, if $k \geq 1$ holds, then $\hat{v}$ is obviously a successor of the node $\hat{v}_{k-1} \in \widehat{V}_{k-1}$ in $\widehat{G}$. It follows that $\widehat{G}$ satisfies both causality constraints required of a semi-run of $(\mathcal{A}')^q$.

(Consistency of $\widehat{L}$) Clearly, $\widehat{L}(\hat{v}_0) = q$ is the initial state of $\mathcal{A}^q$. Let $\hat{e} = \langle \hat{v}_k, \widehat{V}_{k+1}\rangle \in \widehat{E}$ be an edge in $\widehat{G}$ for some $0 \leq k < \ell + 1$. By the definition of $\widehat{L}$, $\widehat{L}(\hat{e}) = t_k = \langle q, \widehat{\Gamma}_k, \widehat{F}_k, \widehat{Q}'_k\rangle$ holds. By the construction of $\widehat{G}$, $\widehat{L}(\hat{v}_k) = q$ holds, and it is easy to check that also $\widehat{L}(\widehat{V}_{k+1}) = \widehat{Q}'_k$ holds. The labeling $\widehat{L}$ is thus consistent if $w^i(k) \in \widehat{\Gamma}_k$ and $t_k \in \Delta \setminus \{t\}$ hold. This is clear if $L(e_k) \neq t$ holds in the original run G, since in this case $t_k = L(e_k) = \langle q, \Gamma_k, F_k, Q'_k\rangle \in \Delta \setminus \{t\}$ holds, and because $e_k \in E \cap (V_{i+k} \times 2^{V_{i+k+1}})$ holds and L is consistent, $w^i(k) \in \Gamma_k = \widehat{\Gamma}_k$ holds. Otherwise, if $L(e_k) = t$ holds, then t_k is one of the $\big\{w^i(k)\big\}$-substitutable transitions $T_{w^i(k)} \subseteq \Delta \setminus \{t\}$, and $w^i(k) \in \widehat{\Gamma}_k$ follows from $\big\{w^i(k)\big\}$-substitutability. It follows that the labeling $\widehat{L}$ is consistent.

($\widehat{G}$ is fin-accepting) If $\ell < \omega$ holds, then $\widehat{E}$ contains only finitely many edges, and thus $\widehat{G}$ is trivially a fin-accepting semi-run of $(\mathcal{A}')^q$ on w^i. Because $\ell \leq j$ holds, this occurs whenever q is a transient state of $\mathcal{A}$: otherwise (if $\ell = j = \omega$), the chain of edges $(e_k)_{0 \leq k < j+1}$ labeled with initial self-loops of $\mathcal{A}^q$ would violate the fin-acceptance condition in the fin-accepting run of $\mathcal{A}^q$ embedded in G.

If $\ell = j = \omega$ holds and q is not a transient state of $\mathcal{A}$, then the run $\widehat{G}$ contains the unique infinite branch $(\hat{e}_k)_{0 \le k < \omega} = \big(\langle \hat{v}_k, \widehat{V}_{k+1} \rangle\big)_{0 \le k < \omega}$. If $L(e_k) = t$ holds for only finitely many $0 \le k < \omega$ in the chain of edges $(e_k)_{0 \le k < \omega}$, then there exists an index $0 \le \ell' < \omega$ such that $\widehat{L}(\hat{e}_k) = L(e_k)$ holds for all $\ell' \le k < \omega$ by the construction of the sequence $(t_k)_{0 \le k < \omega}$ and the definition of the labeling $\widehat{L}$. Because the chain $(e_k)_{0 \le k < \omega}$ is an infinite branch in a fin-accepting run of $\mathcal{A}^q$ on w^i, it follows that $\mathrm{fin}\big((\hat{e}_k)_{0 \le k < \omega}\big) = \mathrm{fin}\big((\hat{e}_k)_{\ell' \le k < \omega}\big) = \mathrm{fin}\big((e_k)_{\ell' \le k < \omega}\big) = \mathrm{fin}\big((e_k)_{0 \le k < \omega}\big) = \emptyset$ holds, and thus $\widehat{G}$ is a fin-accepting semi-run of $(\mathcal{A}')^q$ on w^i.

Otherwise the chain $(e_k)_{0 \le k < \omega}$ contains infinitely many edges labeled with the transition t. We show that for all acceptance conditions $f \in \mathcal{F}$, $\widehat{L}(\hat{e}_k)$ is not an f-transition of $(\mathcal{A}')^q$ for infinitely many $0 \le k < \omega$. Then obviously $f \notin \mathrm{fin}\big((\hat{e}_k)_{0 \le k < \omega}\big)$ holds, and because the same result holds for all acceptance conditions $f \in \mathcal{F}$, we may conclude that $\mathrm{fin}\big((\hat{e}_k)_{0 \le k < \omega}\big) = \emptyset$ holds, and $\widehat{G}$ is fin-accepting. Obviously, this result holds trivially if $\mathcal{F} = \emptyset$ holds.

Assume that $f \in F$ holds, i.e., that the transition t is an f-transition of $\mathcal{A}$. Because $\mathrm{fin}\big((e_k)_{0 \le k < \omega}\big) = \emptyset$ holds, the transition $L(e_k)$ is not an f-transition of $\mathcal{A}$ for infinitely many $0 \le k < \omega$. Because t is an f-transition, however, it follows that $L(e_k) \ne t$ holds at all of these indices. By the definition of the transitions t_k and the semi-run $\widehat{G}$, it follows that $\widehat{L}(\hat{e}_k) = t_k = L(e_k)$ holds at all of these indices, and it is easy to see that $f \notin \mathrm{fin}\big((\hat{e}_k)_{0 \le k < \omega}\big)$ holds.

Let $f \in \mathcal{F} \setminus F$. Because $L(e_k) = t$ holds for infinitely many $0 \le k < \omega$, it follows from the inductive definition of the integers c_k that $f_{c_k} = f$ holds for infinitely many k such that $L(e_k) = t$ holds (the integer c_{k+1} is defined by incrementing c_k modulo the number of acceptance conditions in $\mathcal{F} \setminus F$ iff $L(e_k) = t$ holds). Obviously, the definition of the transitions t_k then guarantees that $t_k = \widehat{L}(\hat{e}_k)$ is not an f-transition of $(\mathcal{A}')^q$ at any of these indices. Therefore $f \notin \mathrm{fin}\big((\hat{e}_k)_{0 \le k < \omega}\big) = \emptyset$ holds also in this case. By the above observation, it follows that $\widehat{G}$ is a fin-accepting semi-run of $(\mathcal{A}')^q$ on w^i.

($\widehat{G}$ can be extended into a fin-accepting run of $(\mathcal{A}')^q$ on w^i) Let $\hat{v} \in \widehat{V}$ be a node with no outgoing edges in $\widehat{G}$. Then $\hat{v} = \hat{v}_{k,k'} \in \widehat{V}_{k+1}$ holds for some $0 \le k < \omega$ and $1 \le k' \le n_k$, and $\widehat{L}(\hat{v}) \ne q$ holds. Because the fin-accepting run G contains the edge $e_k \in V_{i+k} \times 2^{V_{i+k+1}}$, the target nodes of which are labeled with (exactly) the target states of the transition $L(e_k) = \langle q, \Gamma_k, F_k, Q'_k \rangle$, it follows (Proposition 2.3.9) that $(\mathcal{A}^q)^{q'}$ $(= \mathcal{A}^{q'})$ fin-accepts $w^{i+k+1} = (w^i)^{k+1}$ for all $q' \in Q'_k$, i.e., $(w^i)^{k+1} \in \bigcap_{q' \in Q'_k} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^{q'})$ holds.

We show that $(w^i)^{k+1} \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^{\widehat{L}(\hat{v})})$ holds. This is clear if $L(e_k) \ne t$ holds, because in this case $\widehat{L}(\hat{v}) = q_{k,k'} \in Q'_k$ holds by the definition of $\widehat{G}$. Otherwise, if $L(e_k) = t$ holds, the set Q'_k coincides with the set of target states Q' of the transition t, and thus $(w^i)^{k+1} \in \bigcap_{q' \in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^{q'})$ holds. Because the transition $t_k = \langle q, \widehat{\Gamma}_k, \widehat{F}_k, \widehat{Q}'_k \rangle$ is in this case chosen from a set of transitions $\{w^i(k)\}$-substitutable for t under fin-acceptance, it follows that $(w^i)^{k+1} \in \bigcap_{q' \in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^{q'}) \subseteq \bigcap_{q' \in \widehat{Q}'_k} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^{q'})$ holds. In particular, because $\widehat{L}(\hat{v}) \in \widehat{Q}'_k$ holds by the definition of $\widehat{G}$, it follows that $\mathcal{A}^{\widehat{L}(\hat{v})}$ fin-accepts $(w^i)^{k+1}$.

Because $\widehat{L}(\hat{v}) \neq q$ is a successor of q and $\mathcal{A}$ is a self-loop alternating automaton, $\mathcal{A}^{\widehat{L}(\hat{v})} = (\mathcal{A}^q)^{\widehat{L}(\hat{v})} = \big((\mathcal{A}')^q\big)^{\widehat{L}(\hat{v})}$ holds, and it follows that $\big((\mathcal{A}')^q\big)^{\widehat{L}(\hat{v})}$ fin-accepts $(w^i)^{k+1}$. Because the same reasoning applies to all nodes of $\widehat{G}$ with no outgoing edges, $\widehat{G}$ can be extended into a fin-accepting run of $(\mathcal{A}')^q$ on w^i by Proposition 2.3.14.

(G' can be extended into a fin-accepting run of $\mathcal{A}'$ on w) The above construction can now be repeated at every level $0 \leq i < \omega$ of G' containing a node (labeled with the state q) with no outgoing edges to show that $w^i \in \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big)$ holds. By Proposition 2.3.14, it follows that the fin-accepting semi-run G' can be extended into a fin-accepting run of $\mathcal{A}'$ on w. Because the result holds for all $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$, $\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}')$ holds. □

It is easy to see that Proposition 6.2.2 is actually only a special case of Proposition 6.2.3 if the substituted transition is a self-loop of a self-loop alternating automaton simplified in the sense of Corollary 2.3.20. The preconditions of Proposition 6.2.2 are nevertheless simpler than those used in Proposition 6.2.3; additionally, Proposition 6.2.2 is applicable also to substituted transitions that are not self-loops of the automaton.

Other special cases of the results with simpler preconditions can be obtained by strengthening the conditions on substitutability of transitions. For example, instead of finding a $\{\sigma\}$-substitutable transition for every symbol σ in the guard Γ of a substituted transition t, a single non-self-loop transition that is Γ-substitutable for t under fin-acceptance suffices to show in Proposition 6.2.2 that t is redundant. Similarly, a self-loop t can be found to be redundant using Proposition 6.2.3 by exhibiting, instead of a family of sets, a single set of transitions, all of which are Γ-substitutable for t under fin-acceptance and together satisfy the constraint on the intersection of their acceptance conditions. Of course, special cases such as these are less general than the above results and do not apply in as many situations (see Ex. 6.2.11 in Sect. 6.2.6).

6.2.4 Reducing Language Containment Between Intersections of Languages to Language Emptiness

Identically to the language containment tests used in Ch. 5, the general version of the test for the redundancy of a transition in an automaton $\mathcal{A}$ (Sect. 6.1) can be reduced to checking for the emptiness of the set intersection of the language of $\mathcal{A}$ with the complement of the language of another automaton (see Sect. 5.2). Because substitutability under fin-acceptance depends on a condition on language containment between set intersections of two families of languages, applying the standard reduction of language containment into language emptiness to this problem yields an equation which refers to a more general Boolean combination of languages (see below); however, this problem on language emptiness can easily be reduced to multiple tests, each of which again involves only intersections of languages, via basic set-theoretic manipulation as follows.

Lemma 6.2.4 *For any two finite families of languages $\Lambda_1, \Lambda_2 \subseteq 2^{\Sigma^\omega}$ over the finite alphabet Σ, $\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1 \subseteq \bigcap_{\mathcal{L}_2 \in \Lambda_2} \mathcal{L}_2$ holds iff $\big(\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1\big) \cap \overline{\mathcal{L}_2}$ holds* *for all $\mathcal{L}_2 \in \Lambda_2 \setminus \Lambda_1$ (where we write $\overline{\mathcal{L}} = \Sigma^\omega \setminus \mathcal{L}$ for all $\mathcal{L} \subseteq \Sigma^\omega$).*

<u>*Proof:*</u> $\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1 \subseteq \bigcap_{\mathcal{L}_2 \in \Lambda_2} \mathcal{L}_2$ holds

$$
\begin{array}{ll}
\text{iff} & \big(\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1\big) \cap \overline{\bigcap_{\mathcal{L}_2 \in \Lambda_2} \mathcal{L}_2} = \emptyset \\
\text{iff} & \big(\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1\big) \cap \big(\bigcup_{\mathcal{L}_2 \in \Lambda_2} \overline{\mathcal{L}_2}\big) = \emptyset \\
\text{iff} & \bigcup_{\mathcal{L}_2 \in \Lambda_2} \big(\big(\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1\big) \cap \overline{\mathcal{L}_2}\big) = \emptyset \\
\text{iff} & \bigcup_{\mathcal{L}_2 \in \Lambda_2 \setminus \Lambda_1} \big(\big(\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1\big) \cap \overline{\mathcal{L}_2}\big) = \emptyset \quad \big(\big(\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1\big) \cap \overline{\mathcal{L}_2} = \emptyset \textit{ for all } \mathcal{L}_2 \in \Lambda_1\big) \\
\text{iff} & \big(\bigcap_{\mathcal{L}_1 \in \Lambda_1} \mathcal{L}_1\big) \cap \overline{\mathcal{L}_2} = \emptyset \quad \text{holds for all } \mathcal{L}_2 \in \Lambda_2 \setminus \Lambda_1.
\end{array}
$$

□

By instantiating the languages in Lemma 6.2.4 with languages fin-recognized by subautomata of an alternating automaton $\mathcal{A}$, we obtain the following immediate corollary that applies to testing substitutability of transitions under fin-acceptance.

Corollary 6.2.5 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton. For all subsets $Q_1, Q_2 \subseteq Q$,*

$$\bigcap_{q \in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \subseteq \bigcap_{q \in Q_2} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \quad \textit{holds iff} \quad \Big(\bigcap_{q_1 \in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_1})\Big) \cap \overline{\mathcal{L}_{\text{fin}}(\mathcal{A}^{q_2})} = \emptyset$$

holds for all $q_2 \in Q_2 \setminus Q_1$.

A well-known special case (used, for example, by Gastin and Oddoux [2001] in an optimization to their translation procedure from LTL into very weak alternating automata) of the above result arises when the set Q_2 is a subset of the set Q_1. Obviously, the right-hand side condition of Corollary 6.2.5 holds trivially in this case.

Corollary 6.2.6 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton. For all subsets $Q_1, Q_2 \subseteq Q$,*

$$\bigcap_{q \in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \subseteq \bigcap_{q \in Q_2} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \qquad \textit{holds if} \qquad Q_2 \subseteq Q_1.$$

(Intuitively, this result is easy to justify: if $Q_2 \subseteq Q_1$ holds, a word $w \in \bigcap_{q \in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ belongs to all languages $\mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ ($q \in Q_2$) in addition to all languages determined by the states in $Q_1 \setminus Q_2$, and thus it is "harder" for w to belong to the language $w \in \bigcap_{q \in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$.)

6.2.5 Compatibility with Nondeterminization of Automata Built from LTL Formulas

Clearly, an automaton obtained from a self-loop alternating automaton by removing one or more of its redundant transitions is a self-loop alternating automaton. Therefore, this automaton can be translated into a nondeterministic one using the universal subset construction given in Theorem 4.2.1 (Sect. 4.2). In our application of translating LTL formulas into nondeterministic automata, however, we have preferred using the optimized construction of Theorem 4.3.2 (Sect. 4.3.2) instead to avoid introducing new acceptance conditions in the translation. As observed in Sect. 4.3.4, the automata built from LTL formulas using the basic translation rules have special structural and semantic properties (acceptance closed transitions and representative states for acceptance conditions, see Sect. 4.3.3) that allow the automata to be translated into nondeterministic automata using the optimized universal subset construction. These properties are also preserved when applying the new translation rules discussed in Ch. 5. In this section we investigate conditions under which these properties continue to hold when removing redundant initial transitions from self-loop alternating automata.

We first state two results on the preservation of acceptance closure and the existence of representative states for acceptance conditions under the removal of redundant initial transitions from self-loop alternating automata. It is easy to show that any acceptance closed self-loop alternating automaton simplified in the sense of Corollary 2.3.20 remains such an automaton if one of its transitions is removed. (This transition need not even be redundant.)

Proposition 6.2.7 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an acceptance closed self-loop alternating automaton simplified in the sense of Corollary 2.3.20, and let $t \in \Delta$ be a transition of $\mathcal{A}$. Let $\mathcal{A}' \stackrel{def}{=} \langle \Sigma, Q, \Delta \setminus \{t\}, q_I, \mathcal{F} \rangle$ be the automaton obtained from $\mathcal{A}$ by removing the transition t from Δ. The automaton $\mathcal{A}'$ is an acceptance closed self-loop alternating automaton simplified in the sense of Corollary 2.3.20.*

<u>*Proof:*</u> Obviously, the automaton $\mathcal{A}'$ is a self-loop alternating automaton simplified in the sense of Corollary 2.3.20. We check that $\mathcal{A}'$ is acceptance closed. Let $t' \in \Delta \setminus \{t\}$ be an f-transition of $\mathcal{A}'$ for some acceptance condition $f \in \mathcal{F}$. Because the set of acceptance conditions of every non-self-loop transition in Δ is empty, t' is a self-loop of $\mathcal{A}'$, i.e., t' contains its source state in its target states. Clearly, this state is an f-state of $\mathcal{A}'$. Because the same result holds for all transitions and acceptance conditions of $\mathcal{A}'$, it follows that $\mathcal{A}'$ is acceptance closed. □

The following proposition states sufficient conditions under which the removal of a redundant initial transition from an alternating automaton (with no non-self-loop transitions to its initial state) will not interfere with the existence of a representative state for an acceptance condition if the condition still occurs in a transition of the simplified automaton.

Proposition 6.2.8 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton with no non-self-loop transitions having the state q_I as a target state, and let $f \in \mathcal{F}$ be an acceptance condition for which $\mathcal{A}$ has an f-representative state $q_f \in Q$. Let $t = \langle q_I, \Gamma, F, Q' \rangle \in \Delta$ be a redundant initial transition of $\mathcal{A}$ such that there exists, for all $\sigma \in \Gamma$, a nonempty set of transitions $T_\sigma = \{t_{\sigma,1}, t_{\sigma,2}, \ldots, t_{\sigma,n_\sigma}\} \subseteq \Delta \setminus \{t\}$ (for some $1 \leq n_\sigma < \omega$) with acceptance conditions $F_{\sigma,1}, F_{\sigma,2}, \ldots, F_{\sigma,n_\sigma} \subseteq \mathcal{F}$, respectively, such that for all $t' \in T_\sigma$, t' is $\{\sigma\}$-substitutable for t in $\mathcal{A}$ under fin-acceptance, and $\bigcap_{1 \leq i \leq n_\sigma} F_{\sigma,i} \subseteq F$ holds. Let $\mathcal{A}' \stackrel{def}{=} \langle \Sigma, Q, \Delta \setminus \{t\}, \mathcal{F} \rangle$ be the automaton obtained from $\mathcal{A}$ by removing the transition t. The automaton $\mathcal{A}'$ has no non-self-loop transitions having the state q_I as a target state, and either $\mathcal{A}'$ has no f-states, or the state q_f is an f-representative state of $\mathcal{A}'$.*

Proof: Because $\mathcal{A}$ has no non-self-loop transitions having q_I as a target state, it is easy to see from the definition of $\mathcal{A}'$ that $\mathcal{A}'$ has no such transitions, either. If the transition t is the only f-transition of $\mathcal{A}$, then the result follows trivially because $\mathcal{A}'$ has no f-states in this case. Otherwise $\mathcal{A}'$ contains at least one f-state. We need to show that the state q_f is an f-representative state of $\mathcal{A}'$. For this purpose, we first make the following observations:

($\mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big) = \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ *holds for all* $q \in Q$) First, it is easy to see that $\mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big) = \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds for all $q \in Q$: clearly, this equality holds for all $q \in Q \setminus \{q_I\}$, because $(\mathcal{A}')^q = \mathcal{A}^q$ holds in this case (Δ contains no non-self-loop transitions having the state q_I as a target state). Furthermore, the automata $(\mathcal{A}')^{q_I}$ and $\mathcal{A}^{q_I}$ are fin-equivalent, because the transition t is redundant in $\mathcal{A}$.

($\mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^q\big) = \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$ *holds for all* $q \in Q$) We claim that also $\mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^q\big) = \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$ holds for all $q \in Q$. Clearly, $\mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^q\big) \subseteq \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$ holds by the definition of $\mathcal{A}'$ for all $q \in Q$: if the subautomaton $(\mathcal{A}')^q$ is able to fin-accept a word by avoiding an initial f-transition, then so is $\mathcal{A}^q$, since any fin-accepting run of $(\mathcal{A}')^q$ on some $w \in \Sigma^\omega$ is a fin-accepting run of $\mathcal{A}^q$ on w.

To show language containment in the other direction, let $q \in Q$, and let $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$. By Proposition 2.3.15 (and the definition of $\mathcal{L}^f_{\text{fin}}(\mathcal{A}^q)$), there exists a transition $t_q = \langle q, \Gamma_q, F_q, Q'_q\rangle \in \Delta$ such that $w(0) \in \Gamma_q$ and $f \notin F_q$ hold, and $\mathcal{A}^{q'}$ fin-accepts w^1 for all $q' \in Q'_q$. If $t_q \neq t$ holds, then the transition t_q is a transition of $\mathcal{A}'$, and because $w^1 \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) = \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^{q'}\big)$ holds for all $q' \in Q'_q$ by the above observation, it follows (Proposition 2.3.15) that $(\mathcal{A}')^q$ fin-accepts w by avoiding an initial f-transition.

Otherwise $t_q = t$ is the initial transition removed from $\mathcal{A}$. Let $\sigma \overset{def}{=} w(0)$. By the assumption, there exists a nonempty set of transitions $T_\sigma = \{t_{\sigma,1}, t_{\sigma,2}, \ldots, t_{\sigma,n_\sigma}\} \subseteq \Delta \setminus \{t\}$ of $\mathcal{A}$ for some $1 \leq n_\sigma < \omega$ (with acceptance conditions $F_{\sigma,1}, F_{\sigma,2}, \ldots, F_{\sigma,n_\sigma} \subseteq \mathcal{F}$, respectively) such that the transition $t_{\sigma,i}$ is $\{\sigma\}$-substitutable for the transition t in $\mathcal{A}$ under fin-acceptance for all $1 \leq i \leq n_\sigma$, and $\bigcap_{1\leq i\leq n_\sigma} F_{\sigma,i} \subseteq F$ holds. Because the transition $t = t_q$ is not an f-transition, it follows that T_σ contains a transition $t_f = \langle q, \Gamma_f, F_f, Q'_f\rangle \in T_\sigma$ that is not an f-transition of $\mathcal{A}$. Clearly, t_f is an initial transition of the automaton $\mathcal{A}'$, and because t_f is $\{\sigma\}$-substitutable for the transition t in $\mathcal{A}$ under fin-acceptance, it follows that $w(0) \in \Gamma_f$ and $w^1 \in \bigcap_{q'\in Q'_f} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) = \bigcap_{q'\in Q'_f} \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^{q'}\big)$ hold. Therefore, $(\mathcal{A}')^q$ fin-accepts w by avoiding an initial f-transition also in this case. It follows that $\mathcal{L}^f_{\text{fin}}(\mathcal{A}^q) \subseteq \mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^q\big)$ holds for all $q \in Q$.

It is now easy to check that the state q_f has the properties required of an f-representative state of $\mathcal{A}'$. Let $q \in Q$ be an f-state of $\mathcal{A}'$, and let $w \in \Sigma^\omega$ be an infinite word over the alphabet Σ.

($w \in \mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^{q_f}\big) \cap \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big)$ *implies* $w \in \mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^q\big)$) By the above observations, $w \in \mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^{q_f}\big) \cap \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big)$ implies that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^{q_f}) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ holds. Because q_f is an f-representative state of $\mathcal{A}$, it follows that $w \in \mathcal{L}^f_{\text{fin}}(\mathcal{A}^q) = \mathcal{L}^f_{\text{fin}}\big((\mathcal{A}')^q\big)$ holds.

<u>*($w \in \mathcal{L}^{f}_{\text{fin}}\big((\mathcal{A}')^{q}\big)$ implies $w^i \in \mathcal{L}^{f}_{\text{fin}}\big((\mathcal{A}')^{q_f}\big)$ for some i)*</u> Assume that $w \in \mathcal{L}^{f}_{\text{fin}}\big((\mathcal{A}')^{q}\big)$ $(= \mathcal{L}^{f}_{\text{fin}}(\mathcal{A}^{q}))$ holds. Because q_f is an f-representative state of $\mathcal{A}$, it follows that there exists an index $0 \leq i < \omega$ such that $w^i \in \mathcal{L}^{f}_{\text{fin}}(\mathcal{A}^{q_f}) = \mathcal{L}^{f}_{\text{fin}}\big((\mathcal{A}')^{q_f}\big)$ holds.

Clearly, the same reasoning applies to all f-states of $\mathcal{A}'$. We conclude that q_f is an f-representative state of $\mathcal{A}'$. □

Proposition 6.2.8 characterizes a set of sufficient conditions under which it is safe to remove a redundant initial transition from an automaton without invalidating the existence of representative states for the acceptance conditions occurring in the transitions of the automaton. It is easy to see that these conditions hold whenever using Proposition 6.2.2 to detect redundant transitions in self-loop automata simplified in the sense of Corollary 2.3.20; a similar result holds for a strengthened version of Proposition 6.2.3 that does not treat transient states of the automaton as a special case.

Proposition 6.2.9 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a self-loop alternating automaton simplified in the sense of Corollary 2.3.20. Let $t = \langle q_I, \Gamma, F, Q' \rangle \in \Delta$ be a transition of $\mathcal{A}$ that can be found to be redundant in $\mathcal{A}$ by applying Proposition 6.2.2, or Proposition 6.2.3, without considering transient states of $\mathcal{A}$ as a special case. For all $\sigma \in \Gamma$, there exists a nonempty set of transitions $T_\sigma = \{t_{\sigma,1}, t_{\sigma,2}, \ldots, t_{\sigma,n_\sigma}\}$ (for some $1 \leq n_\sigma < \omega$) with acceptance conditions $F_{\sigma,1}, F_{\sigma,2}, \ldots, F_{\sigma,n_\sigma} \subseteq \mathcal{F}$, respectively, such that for all $t' \in T_\sigma$, t' is $\{\sigma\}$-substitutable for t in $\mathcal{A}$ under fin-acceptance, and $\bigcap_{1 \leq i \leq n_\sigma} F_{\sigma,i} \subseteq F$ holds.*

<u>*Proof:*</u> If the transition t can be found to be redundant in $\mathcal{A}$ by applying Proposition 6.2.2, $\mathcal{A}$ has (for all $\sigma \in \Gamma$) a non-self-loop initial transition $t_\sigma \in \Delta \setminus \{t\}$ that is $\{\sigma\}$-substitutable for t in $\mathcal{A}$ under fin-acceptance. Because the transition t_σ is not a self-loop, it has an empty set of acceptance conditions ($\mathcal{A}$ is simplified in the sense of Corollary 2.3.20), and thus the set $\{t_\sigma\}$ satisfies the conditions required of the set T_σ.

Otherwise, if t is found to be redundant by applying Proposition 6.2.3 (without considering transient states of $\mathcal{A}$ as a special case), the existence of the family $\{T_\sigma\}_{\sigma \in \Gamma}$ follows directly from the conditions of Proposition 6.2.3. □

If $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F} \rangle$ is a self-loop alternating automaton built from an LTL formula $\varphi \in LTL^{\text{PNF}}(AP)$ using the (basic or refined) translation rules, then $\mathcal{A}$ is an acceptance closed self-loop alternating automaton with an f-representative state for all acceptance conditions $f \in \mathcal{F}$ for which it has an f-state (Lemma 4.3.8, Proposition 5.4.3, Proposition 5.5.1). Furthermore, because all translation rules guarantee that the set of acceptance conditions of every non-self-loop transition of $\mathcal{A}$ is empty, $\mathcal{A}$ is constructed simplified in the sense of Corollary 2.3.20. Because no automaton built using the translation rules obviously has any non-self-loop transitions to its initial state, either, $\mathcal{A}$ satisfies the assumptions used in Proposition 6.2.7 and Proposition 6.2.8 (for all acceptance conditions $f \in \mathcal{F}$). Therefore, all of these properties are preserved if one (or more) of the redundant initial transitions of $\mathcal{A}$ is removed, provided that each of these transitions satisfies the constraints given in Proposition 6.2.8. (By Proposition 6.2.9, these conditions are guaranteed to hold for every transition that can be found to be redundant using Proposition 6.2.2 or a version of Proposition 6.2.3 strengthened by removing the special case for transient states of the automaton.) As noted in Corollary 4.3.7, the simplified automaton can be translated into a fin-equivalent nondeterministic automaton using the optimized universal subset construction of Theorem 4.3.2 without introducing new acceptance conditions; additionally, the simplified automaton can be used again in a further application of a translation rule.

6.2.6 Examples

In this section we illustrate the simplification of self-loop alternating automata using Proposition 6.2.2 and Proposition 6.2.3 with several examples.

A Final Look at the Running Example
We first reconsider automata built in Ex. 5.6.1 (Sect. 5.6).

Example 6.2.10 As seen in Ex. 5.6.1, the LTL formula

$$\varphi \stackrel{\text{def}}{\equiv} \Big(\big(\bot \mathsf{R_w} (\top \mathsf{U_s} p_1)\big) \wedge \big(\bot \mathsf{R_w} (\top \mathsf{U_s} p_2)\big) \Big) \in LTL\big(\{p_1, p_2\}\big)$$

can be translated into a single-state automaton (repeated in Fig. 6.2 (a)) using the refined translation rules defined in Sect. 5.5 (Table 5.3). Let q denote the unique state of this automaton. The transition relation of the automaton consists of the self-loops $t_1 = \big\langle q, \top, \{\bullet, \circ\}, \{q\}\big\rangle$, $t_2 = \big\langle q, p_1, \{\circ\}, \{q\}\big\rangle$, $t_3 = \big\langle q, p_2, \{\bullet\}, \{q\}\big\rangle$ and $t_4 = \big\langle q, (p_1 \wedge p_2), \emptyset, \{q\}\big\rangle$.

Clearly, the guard of the transition t_4 encodes the unique symbol $\{p_1, p_2\}$ of the alphabet $2^{\{p_1,p_2\}}$ of the automaton. Because $\big((p_1 \wedge p_2) \rightarrow p_1\big)$ and $\big((p_1 \wedge p_2) \rightarrow p_2\big)$ are valid propositional implications (i.e., $\big\{\{p_1, p_2\}\big\} \subseteq \big\{\{p_1\}, \{p_1, p_2\}\big\}$ and $\big\{\{p_1, p_2\}\big\} \subseteq \big\{\{p_2\}, \{p_1, p_2\}\big\}$ hold) and because t_2, t_3 and t_4 share their target states, the transitions t_2 and t_3 are $\{p_1, p_2\}$-substitutable under fin-acceptance for t_4 in the automaton. Because the intersection of the acceptance conditions of t_2 and t_3 is empty, it follows that the set of transitions $\{t_2, t_3\}$ satisfies the preconditions of Proposition 6.2.3 on the unique symbol in the guard of t_4. Therefore the transition t_4 is redundant, and it can be removed from the automaton to obtain the automaton shown in Fig. 6.2 (b).

Joining the automaton in Fig. 6.2 (b) with the automaton in Fig. 6.2 (c) (built for the formula $\psi \stackrel{\text{def}}{\equiv} \big(p_3 \mathsf{R_w} (p_4 \mathsf{R_s} p_5)\big)$ in Sect. 5.6) using the translation rule for the $\vee$ connective yields the automaton shown in Fig. 6.2 (d) for the formula $(\varphi \vee \psi)$. We can now apply Proposition 6.2.2 several times to remove some of the initial transitions of this automaton: for example, because $(p_2 \rightarrow \top)$ is a valid propositional implication, the $\top$-labeled initial non-self-loop of the automaton is $\{\sigma\}$-substitutable for the initial transition with guard p_2 (under fin-acceptance, because the transitions share their target state) for all $\sigma \in 2^{\{p_1,p_2\}}$ such that $\sigma \models p_2$ holds, and thus the latter transition can be removed from the automaton by Proposition 6.2.2. Two further applications of this result then yield the automaton shown in Fig. 6.2 (e). ■

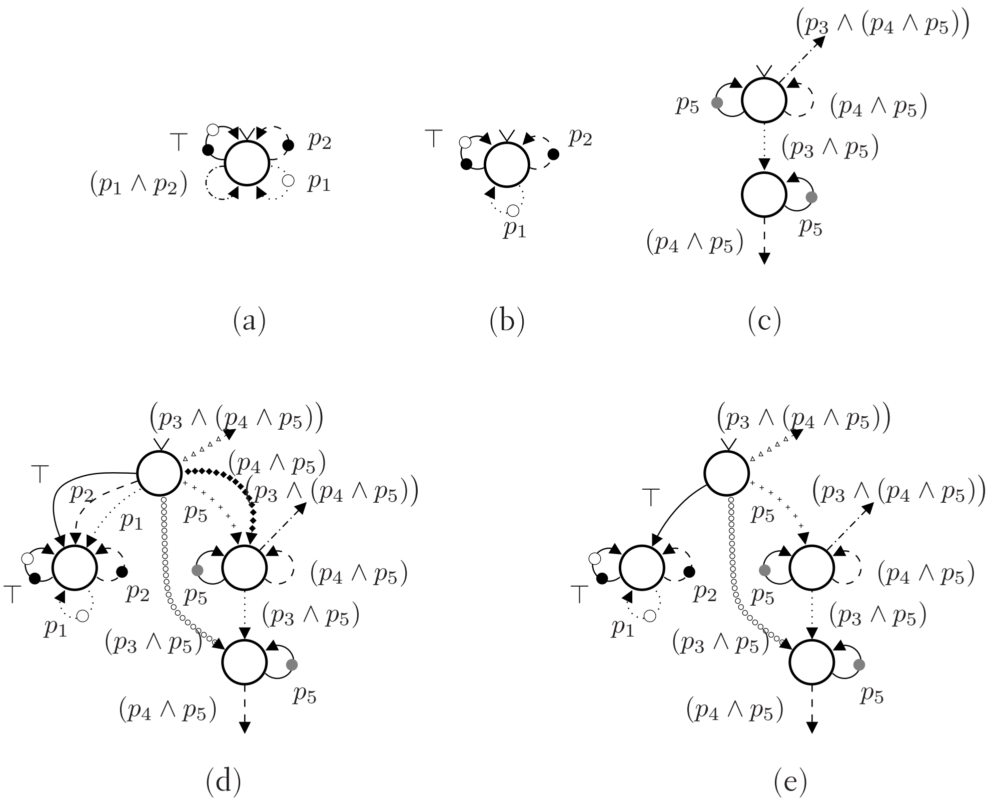


Fig. 6.2: Using Proposition 6.2.2 and Proposition 6.2.3 to optimize translation of LTL formulas into self-loop alternating automata. (a) Automaton built for the formula $\varphi \stackrel{def}{\equiv} \Big((\perp \mathsf{R_w}\, (\top\, \mathsf{U_s}\, p_1)) \wedge (\perp \mathsf{R_w}\, (\top\, \mathsf{U_s}\, p_2)) \Big)$ in Sect. 5.6; (b) Automaton obtained from (a) by applying Proposition 6.2.3; (c) Automaton built for the formula $\psi \stackrel{def}{\equiv} (p_3\, \mathsf{R_w}\, (p_4\, \mathsf{R_s}\, p_5))$ in Sect. 5.6; (d) Automaton built from (b) and (c) for the formula $(\varphi \vee \psi)$; (e) Automaton obtained from (d) by repeated application of Proposition 6.2.2

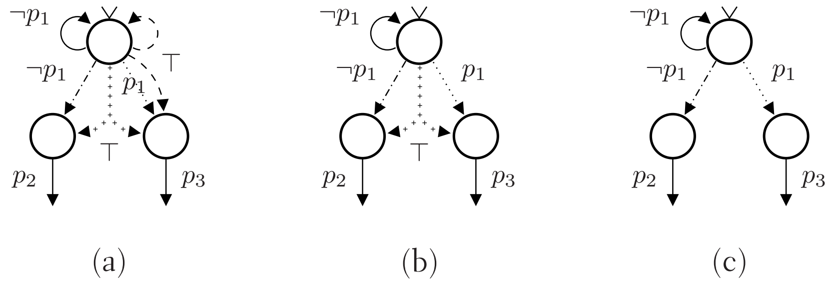


Fig. 6.3: Using Proposition 6.2.2 and Proposition 6.2.3 to find redundant initial transitions that cannot be detected using the stronger notion of substitutability mentioned at the end of Sect. 6.2.3. (a) A self-loop alternating automaton; (b) Automaton obtained from (a) by applying Proposition 6.2.3; (c) Automaton obtained from (b) by applying Proposition 6.2.2

Comparison between Notions of Transition Substitutability

As a matter of fact, the previous example remains the same when using the stronger notion of substitutability (i.e., Γ-substitutability with respect to the guard Γ of a transition instead of $\{\sigma\}$-substitutability on individual symbols $\sigma \in \Gamma$) mentioned at the end of Sect. 6.2.3 (p. 167) when applying Proposition 6.2.2 and Proposition 6.2.3. In some cases, however, the weaker notion of substitutability is necessary to detect the redundancy of a transition using Proposition 6.2.2 or Proposition 6.2.3.

Example 6.2.11 Consider the self-loop alternating automaton (with the alphabet $2^{\{p_1,p_2,p_3\}}$) shown in Fig. 6.3 (a). (This automaton can be obtained by translating the LTL formula $\big((p_1 \vee \mathsf{X}p_2)\,\mathsf{R_w}\,(\neg p_1 \vee \mathsf{X}p_3)\big)$ into a self-loop alternating automaton and then simplifying the guards of its transitions as described in Sect. 5.1.2; the basic and the refined translation rules yield the same automaton for this LTL formula.) Denote the initial state of the automaton by q_I, and denote the two other states of the automaton (from left to right) by q_1 and q_2. The initial transitions of the automaton can then be written as $t_1 = \big\langle q_I, \neg p_1, \emptyset, \{q_I\}\big\rangle$ (the initial self-loop of the automaton with guard $\neg p_1$), $t_2 = \big\langle q_I, \top, \emptyset, \{q_I, q_2\}\big\rangle$ (the other initial self-loop), $t_3 = \big\langle q_I, p_1, \emptyset, \{q_2\}\big\rangle$ (the transition with guard p_1), $t_4 = \big\langle q_I, \neg p_1, \emptyset, \{q_1\}\big\rangle$ (the non-self-loop transition with guard $\neg p_1$) and $t_5 = \big\langle q_I, \top, \emptyset, \{q_1, q_2\}\big\rangle$.

Clearly, the guard Γ_2 of the transition t_2 contains all symbols of the automaton's alphabet (i.e., $\Gamma_2 = 2^{\{p_1,p_2,p_3\}}$ holds). It is easy to see that the transition t_1 is $\{\sigma\}$-substitutable for the transition t_2 for all $\sigma \in \Gamma_2$ such that $p_1 \notin \sigma$ holds. Because the transition t_3 is substitutable for t_2 on the remaining symbols in Γ_2 and because the target state sets of t_1 and t_3 are subsets of the target state set of the transition t_2, it follows that for all $\sigma \in \Gamma_2$, one of the transitions t_1 and t_3 is $\{\sigma\}$-substitutable for t_2 under fin-acceptance (the condition on fin-acceptance follows directly by Corollary 6.2.6). Because the automaton has no acceptance conditions, it is easy to see that there exists (for all symbols $\sigma \in \Gamma_2$) a set of transitions ($\{t_1\}$ or $\{t_3\}$) that satisfies the conditions required in Proposition 6.2.3 (using the self-loop t_2 as the substituted transition). Therefore the transition t_2 is redundant; removing this transition from the automaton yields the automaton shown in Fig. 6.3 (b). (Note that the redundancy of t_2 cannot be shown with Proposition 6.2.3 using the stronger notion of Γ_2-substitutability, because neither t_1 nor t_3 is Γ_2-substitutable for t_2 in $\mathcal{A}$.)

Similarly, it is straightforward to check that the transition t_3 is $\{\sigma\}$-substitutable under fin-acceptance also for the transition t_5 for all $\sigma \in 2^{\{p_1,p_2,p_3\}}$ such that $p_1 \in \sigma$ holds. Because the transition t_4 is $\{\sigma\}$-substitutable under fin-acceptance for t_5 on the remaining symbols in the guard of t_5, it follows (by Proposition 6.2.2, because neither of the transitions t_3 or t_4 is a self-loop) that also the transition t_5 is redundant and can be removed from the automaton as shown in Fig. 6.3 (c). (As above, neither t_3 nor t_4 is substitutable by itself for t_5 on all symbols in the guard of t_5.) ■

Comparison with the Translation of Gastin and Oddoux [2001]

The following examples demonstrate cases in which the refined translation rules—combined with transition removal using Proposition 6.2.3 using a restricted notion of transition substitutability suggested by Gastin and Oddoux [2001]—yield automata with a polynomial number of transitions for certain LTL formulas for which this notion of substitutability does not allow any simplification of the automata built for the same formulas using the basic rules, leading to an exponential blow-up in the number of transitions instead. Because the basic translation rules given in Sect. 3.1 are essentially identical to the ones used by Gastin and Oddoux [2001] in their translation, we obtain in this way a comparison of our refined translation against their translation procedure. Incidentally, the families of formulas in the examples are virtually the same as those used by Gastin and Oddoux [2001] for benchmarking their implementation against other implementations.

For convenience, we shall write guards of transitions using set notation: for a formula $\theta \in PL(AP)$, we write $\Gamma_\theta \subseteq 2^{AP}$ as the set of models of θ; obviously, $\Gamma_\top \cap \Gamma = \Gamma$ holds for all $\Gamma \subseteq 2^{AP}$. Additionally, we say that a transition $t' = \langle q', \Gamma', F', Q'' \rangle$ of a self-loop alternating automaton is substitutable for another transition $t = \langle q, \Gamma, F, Q' \rangle$ of the automaton *in the sense of Gastin and Oddoux* iff t' is Γ-substitutable for t in the usual sense (i.e., if $t' \neq t$, $q' = q$ and $\Gamma \subseteq \Gamma'$ hold), and $Q'' \subseteq Q'$ holds. (By Corollary 6.2.6, it is immediate that substitutability in the sense of Gastin and Oddoux implies substitutability under fin-acceptance.)

Example 6.2.12 Let $\{\varphi_n\}_{1 \leq n < \omega}$ be the family of LTL formulas over the set of atomic propositions $AP = \{p_1, p_2, p_3, \ldots\}$ defined inductively by the rules

$$\begin{aligned} \varphi_1 &\stackrel{def}{=} \mathsf{GF}p_1, \text{ and} \\ \varphi_{i+1} &\stackrel{def}{=} (\varphi_i \wedge \mathsf{GF}p_i) \text{ for all } 1 \leq i < \omega, \end{aligned}$$

where, as usual, $\mathsf{GF}p_i$ is a shorthand for the formula $\big(\bot \mathsf{R_w} (\top \mathsf{U_s}\, p_i)\big)$ for all $1 \leq i < \omega$. Formulas in this family appear in formulas of the form $\neg\big((\bigwedge_{i=1}^{n} \mathsf{GF}p_i) \rightarrow \psi\big)$ (for a fixed formula ψ) used by Gastin and Oddoux [2001] in their benchmarks.

<u>*(Translation using the basic rules)*</u> It is easy to check that the automaton built for a formula $\mathsf{GF}p_i$ for any $1 \leq i < \omega$ using the basic translation rules has two initial transitions

$$\big\{\big\langle q_{\mathsf{GF}p_i}, \Gamma_\top, \emptyset, \{q_{\mathsf{GF}p_i}, q_{\mathsf{F}p_i}\}\big\rangle, \big\langle q_{\mathsf{GF}p_i}, \Gamma_{p_i}, \emptyset, \{q_{\mathsf{GF}p_i}\}\big\rangle\big\}$$

(where we write q_ψ for the initial state of the automaton built for the formula ψ); see Fig. 3.2 (c) on p. 46 for illustration. It is easy to see that neither of these transitions is substitutable for the other in the sense of Gastin and Oddoux since neither $\Gamma_\top \subseteq \Gamma_{p_i}$ nor $\{q_{\mathsf{GF}p_i}, q_{\mathsf{F}p_i}\} \subseteq \{q_{\mathsf{F}p_i}\}$ hold.

Let $1 \leq i < \omega$. Assume that every initial transition of the automaton $\mathcal{A}_{\varphi_i}$ built for the formula φ_i has an empty set of acceptance conditions and a nonempty guard Γ such that for all $\sigma \in 2^{AP}$ and $j > i$, $\sigma \in \Gamma$ holds only if both $\sigma \cup \{p_j\} \in \Gamma$ and $\sigma \setminus \{p_j\} \in \Gamma$ hold (i.e., the guard does not "depend" on the proposition p_j). Assume also that no initial transition of the automaton $\mathcal{A}_{\varphi_i}$ built for the formula φ_i is substitutable for another one of these transitions in the sense of Gastin and Oddoux. Therefore none of these transitions is redundant under this notion of substitutability. It is easy to check that all of these properties hold for the initial transitions of the automaton $\mathcal{A}_{\varphi_1}$.

Applying the basic translation rule for the $\wedge$ connective (Table 3.1) to the automata built for the formulas φ_i and $\mathsf{GF}p_{i+1}$ yields an automaton $\mathcal{A}_{\varphi_{i+1}}$ having the initial transitions

$$\begin{aligned}&\big\{\big\langle q_{\varphi_{i+1}}, \Gamma, \emptyset, Q' \cup \{q_{\mathsf{GF}p_{i+1}}, q_{\mathsf{F}p_{i+1}}\}\big\rangle \,\big|\\ &\qquad\qquad\qquad \langle q_{\varphi_i}, \Gamma, \emptyset, Q'\rangle \text{ is an initial transition of } \mathcal{A}_{\varphi_i}\big\}\\ \cup\, &\big\{\big\langle q_{\varphi_{i+1}}, \Gamma \cap \Gamma_{p_{i+1}}, \emptyset, Q' \cup \{q_{\mathsf{GF}p_{i+1}}\}\big\rangle \,\big|\\ &\qquad\qquad\qquad \langle q_{\varphi_i}, \Gamma, \emptyset, Q'\rangle \text{ is an initial transition of } \mathcal{A}_{\varphi_i}\big\}.\end{aligned}$$

Denote the two components in this set union by $\Delta_{i+1,1}$ and $\Delta_{i+1,2}$, respectively. Clearly, the set of acceptance conditions of all initial transitions of $\mathcal{A}_{\varphi_{i+1}}$ is empty, and by the induction hypothesis, it is easy to see that every initial transition of $\mathcal{A}_{\varphi_{i+1}}$ has a nonempty guard Γ such that for all $j > i+1$ and $\sigma \in 2^{AP}$, $\sigma \in \Gamma$ holds only if both $\sigma \cup \{p_j\}$ and $\sigma \setminus \{p_j\}$ hold.

Suppose that there exists a pair of transitions $t_1 = \langle q_{\varphi_{i+1}}, \Gamma_1, \emptyset, Q'_1\rangle$, $t_2 = \langle q_{\varphi_{i+1}}, \Gamma_2, \emptyset, Q'_2\rangle \in \Delta_{i+1,1} \cup \Delta_{i+1,2}$ such that the transition t_2 is substitutable for the transition t_1 in the sense of Gastin and Oddoux. Then $t_1 \neq t_2$, $\Gamma_1 \subseteq \Gamma_2$, and $Q'_2 \subseteq Q'_1$ hold. From the above definition of the initial transitions of $\mathcal{A}_{\varphi_{i+1}}$ it follows that the automaton $\mathcal{A}_{\varphi_i}$ has a pair of initial transitions $\hat{t}_1 = \langle q_{\varphi_i}, \widehat{\Gamma}_1, \emptyset, \widehat{Q}_1\rangle$ and $\hat{t}_2 = \langle q_{\varphi_i}, \widehat{\Gamma}_2, \emptyset, \widehat{Q}_2\rangle$ for some $\widehat{\Gamma}_1, \widehat{\Gamma}_2 \subseteq 2^{AP}$ and $\widehat{Q}_1, \widehat{Q}_2 \subseteq Q_{\varphi_i}$ (the state set of $\mathcal{A}_{\varphi_i}$, which contains neither $q_{\mathsf{GF}p_{i+1}}$ nor $q_{\mathsf{F}p_{i+1}}$). There are the following cases:

<u>*($t_1 \in \Delta_{i+1,1}$, $t_2 \in \Delta_{i+1,2}$)*</u> In this case $\Gamma_1 = \widehat{\Gamma}_1$ and $\Gamma_2 = \widehat{\Gamma}_2 \cap \Gamma_{p_{i+1}}$. By the induction hypothesis, there exists a $\sigma \in 2^{AP}$ such that $\sigma \in \widehat{\Gamma}_1 = \Gamma_1$ holds, and thus also $\sigma \setminus \{p_{i+1}\} \in \Gamma_1$ holds. But then it cannot be the case that $\Gamma_1 \subseteq \Gamma_2$ ($\subseteq \Gamma_{p_{i+1}}$) holds, contradictory to the assumption that the transition t_2 is substitutable for the transition t_1.

<u>*($t_1 \in \Delta_{i+1,2}$, $t_2 \in \Delta_{i+1,1}$)*</u> This case is impossible, because $q_{\mathsf{F}p_{i+1}} \in Q'_2 \setminus Q'_1$, and therefore Q'_2 cannot be a subset of Q'_1.

<u>*($t_1, t_2 \in \Delta_{i+1,j}$ for some $j \in \{1,2\}$)*</u> In this case either $\widehat{\Gamma}_1 = \Gamma_1$ and $\widehat{\Gamma}_2 = \Gamma_2$ ($j = 1$), or $\widehat{\Gamma}_1 = \Gamma_1 \cup \big\{\sigma \setminus \{p_{i+1}\} \,\big|\, \sigma \in \Gamma_1\big\}$ and $\widehat{\Gamma}_2 = \Gamma_2 \cup \big\{\sigma \setminus \{p_{i+1}\} \,\big|\, \sigma \in \Gamma_2\big\}$ ($j = 2$), and $\widehat{Q}_1 = Q'_1 \setminus \widehat{Q}$ and $\widehat{Q}_2 = Q'_2 \setminus \widehat{Q}$ for some nonempty set $\widehat{Q} \subseteq \{q_{\mathsf{GF}p_{i+1}}, q_{\mathsf{F}p_{i+1}}\}$. (The transitions $\hat{t}_1$ and $\hat{t}_2$ are necessarily distinct, since otherwise it would be the case that $t_1 = t_2$).

But then, because $\Gamma_1 \subseteq \Gamma_2$ and $Q'_2 \subseteq Q'_1$ hold, it follows that $\widehat{\Gamma}_1 \subseteq \widehat{\Gamma}_2$ and $\widehat{Q}_2 \subseteq \widehat{Q}_1$, contradictory to the assumption that no initial transition of the automaton $\mathcal{A}_{\varphi_i}$ is substitutable for another in the sense of Gastin and Oddoux.

It follows that $\Delta_{i+1,1} \cup \Delta_{i+1,2}$ contains no pairs of initial transitions, one of which would be substitutable for the other in the sense of Gastin and Oddoux; by induction, this result holds for all $1 \leq i < \omega$. Because the automaton $\mathcal{A}_{\varphi_i}$ has two initial transitions for all $1 \leq i < \omega$, it follows (by an argument analogous to the one used in Ex. 3.2.4 in Sect. 3.2.2) that the automaton $\mathcal{A}_{\varphi_i}$ has 2^i initial transitions for all $1 \leq i < \omega$.

(Translation using the refined rules) As observed in Sect. 5.6.3, every formula φ_i $(1 \leq i < \omega)$ belongs to a subclass of formulas that can be translated into a single-state automaton using the refined translation rules (see, for example, Fig. 5.8 (c) and Fig. 5.8 (d) on page 146 for the cases $i = 1$ and $i = 2$, respectively). However, it is clear from the definition of the refined translation rule for the $\wedge$ connective (Table 5.2) that the automaton built for the formula φ_i will still have an exponential number of initial transitions in i if no redundant transitions can be removed from the automaton.

The formula φ_2 has previously appeared as a subformula in our running example considered in Ex. 3.1.1, Ex. 5.6.1 and Ex. 6.2.10. In the last of these examples, we used Proposition 6.2.3 to eliminate one of the initial transitions of the automaton built for the formula φ_2 (see Fig. 6.2 (a) and Fig. 6.2 (b)). As we now show, this simplification is a special case of a general pattern concerning the construction of an automaton for the formula φ_{i+1} from the automaton built for the formulas φ_i and $\mathsf{GF}p_{i+1}$.

Suppose that $\mathcal{A}_{\varphi_i} = \langle 2^{AP}, \{q_{\varphi_i}\}, \Delta_{\varphi_i}, q_{\varphi_i}, \mathcal{F}_{\varphi_i} \rangle$ $(1 \leq i < \omega)$ is a single-state automaton built for the formula φ_i with initial state q_{φ_i}, i acceptance conditions $\mathcal{F}_{\varphi_i} = \{f_1, f_2, \ldots, f_i\}$, and $i+1$ initial self-loop transitions $\Delta_{\varphi_i} = \{t_0, t_1, \ldots, t_i\}$, where $t_0 = \langle q_{\varphi_i}, \Gamma_\top, \mathcal{F}_{\varphi_i}, \{q_{\varphi_i}\} \rangle$, and $t_j = \langle q_{\varphi_i}, \Gamma_{p_j}, \mathcal{F}_{\varphi_i} \setminus \{f_j\}, \{q_{\varphi_i}\} \rangle$ for all $1 \leq j \leq i$. Clearly, the automaton $\mathcal{A}_{\varphi_1}$ built for the formula φ_1 using the refined translation rules is of this form (cf. Fig. 5.8 (c)). We show that joining the automaton $\mathcal{A}_{\varphi_i}$ with an automaton built for the formula $\mathsf{GF}p_{i+1}$ using the refined rule for the $\wedge$ connective (Table 5.2) yields an automaton which can be simplified to the form $\mathcal{A}_{\varphi_{i+1}}$ by repeatedly applying Proposition 6.2.3 to remove some of its initial transitions, using the notion of transition substitutability in the sense of Gastin and Oddoux. By induction, it then follows that the automaton built in this way for the formula φ_i will have $i+1$ instead of 2^i (initial) transitions for all $1 \leq i < \omega$. (It can be checked that Δ_{φ_i} contains no redundant transitions.)

Clearly, the automaton built for the formula $\mathsf{GF}p_{i+1}$ using the refined translation rules is structurally identical to the automaton $\mathcal{A}_{\varphi_1}$ except for the guards and acceptance conditions associated with its transitions: more precisely, the transition relation of this automaton consists of two initial self-loop transitions with guards $\Gamma_\top$ and $\Gamma_{p_{i+1}}$, respectively, such that the transition with guard $\Gamma_\top$ has also an associated acceptance condition f_{i+1} (not included in $\mathcal{F}_{\varphi_i}$, the acceptance conditions of the automaton $\mathcal{A}_{\varphi_i}$). Applying the refined $\wedge$ translation rule (Table 5.2) to the automaton $\mathcal{A}_{\varphi_i}$ and this automaton yields an automaton $\widehat{\mathcal{A}}_{\varphi_{i+1}} = \langle 2^{AP}, \{q_{\varphi_{i+1}}\}, \widehat{\Delta}_{\varphi_{i+1}}, q_{\varphi_{i+1}}, \mathcal{F}_{\varphi_i} \cup \{f_{i+1}\} \rangle$, where

$$\begin{aligned}\widehat{\Delta}_{\varphi_{i+1}} = {} & \Big\{\big\langle q_{\varphi_{i+1}}, \Gamma_\top, \mathcal{F}_{\varphi_i} \cup \{f_{i+1}\}, \{q_{\varphi_{i+1}}\}\big\rangle, \underbrace{\big\langle q_{\varphi_{i+1}}, \Gamma_{p_{i+1}}, \mathcal{F}_{\varphi_i}, \{q_{\varphi_{i+1}}\}\big\rangle}_{t_{i+1,0}}\Big\} \\ & \cup \Big\{\underbrace{\big\langle q_{\varphi_{i+1}}, \Gamma_{p_j}, (\mathcal{F}_{\varphi_i} \setminus \{f_j\}) \cup \{f_{i+1}\}, \{q_{\varphi_{i+1}}\}\big\rangle}_{t_{i+1,j}} \Big| 1 \le j \le i\Big\} \\ & \cup \Big\{\underbrace{\big\langle q_{\varphi_{i+1}}, \Gamma_{p_j} \cap \Gamma_{p_{i+1}}, \mathcal{F}_{\varphi_i} \setminus \{f_j\}, \{q_{\varphi_{i+1}}\}\big\rangle}_{t'_{i+1,j}} \Big| 1 \le j \le i\Big\}.\end{aligned}$$

Let $1 \le j \le i$. Comparing the guard of the transition $t'_{i+1,j}$ with the guard of $t_{i+1,0}$ and $t_{i+1,j}$, we see that $\Gamma_{p_j} \cap \Gamma_{p_{i+1}} \subseteq \Gamma_{p_{i+1}}$ and $\Gamma_{p_j} \cap \Gamma_{p_{i+1}} \subseteq \Gamma_{p_j}$ hold. Therefore, because all transitions in $\widehat{\Delta}_{\varphi_{i+1}}$ share their set of target states, it follows that the transitions $t_{i+1,0}$ and $t_{i+1,j}$ are substitutable for the transition $t'_{i+1,j}$ in the sense of Gastin and Oddoux. Furthermore, because the intersection of the acceptance conditions of $t_{i+1,0}$ and $t_{i+1,j}$ equals $\mathcal{F}_{\varphi_i} \setminus \{f_j\}$ (that is, the set of acceptance conditions of $t'_{i+1,j}$), the transition $t'_{i+1,j}$ satisfies the requirements of Proposition 6.2.3 (using the set $\{t_{i+1,0}, t_{i+1,j}\}$ as the set T_σ for all $\sigma \in \Gamma_{p_j} \cap \Gamma_{p_{i+1}}$). Therefore, the transition $t'_{i+1,j}$ may be removed from $\widehat{\Delta}_{\varphi_{i+1}}$. By repeating the same consideration for all $1 \le j \le i$, we obtain the transition relation

$$\begin{aligned}\widehat{\Delta}'_{\varphi_{i+1}} = {} & \big\{\big\langle q_{\varphi_{i+1}}, \Gamma_\top, \mathcal{F}_{\varphi_{i+1}}, \{q_{\varphi_{i+1}}\}\big\rangle\big\} \\ & \cup \big\{\big\langle q_{\varphi_{i+1}}, \Gamma_{p_j}, \mathcal{F}_{\varphi_{i+1}} \setminus \{f_j\}, \{q_{\varphi_{i+1}}\}\big\rangle \big| 1 \le j \le i+1\big\},\end{aligned}$$

which is of the desired form $\Delta_{\varphi_{i+1}}$ if we define $\mathcal{F}_{\varphi_{i+1}} \overset{def}{=} \mathcal{F}_{\varphi_i} \cup \{f_{i+1}\}$. ■

Example 6.2.13 Let $\{\varphi_n\}_{2 \le n < \omega}$ be a family of LTL formulas over the set of atomic propositions $AP = \{p_1, p_2, p_3, \ldots\}$ defined inductively by the rules

$$\begin{aligned}\varphi_2 & \overset{def}{=} (p_2 \mathsf{R_w}\, p_1), \text{ and} \\ \varphi_{i+1} & \overset{def}{=} (p_{i+1} \mathsf{R_w}\, \varphi_i) \text{ for all } 2 \le i < \omega.\end{aligned}$$

(Again, this family of LTL formulas is essentially identical to a family of formulas of the form $\neg\big(p_1 \mathsf{U_s}\, (p_2 \mathsf{U_s}\, (\ldots \mathsf{U_s}\, p_n) \ldots)\big)$ considered previously by Gastin and Oddoux [2001]; obviously, the positive normal form of any formula of this form can be identified with the formula φ_n by renaming its literals.)

For all $1 \le i < \omega$, let q_{p_i} and $\Delta^I_{p_i}$ be the initial state and the set of initial transitions (respectively) of the alternating automaton built for the atomic proposition $p_i \in AP$ using the translation rules. Clearly, $\Delta^I_{p_i} = \big\{\langle q_{p_i}, \Gamma_{p_i}, \emptyset, \emptyset\rangle\big\}$ for all $1 \le i < \omega$.

(Translation using the basic rules) Let q_{φ_i} and $\Delta^I_{\varphi_i}$ denote the initial state and the set of initial transitions (respectively) of the automaton built for the formula φ_i ($2 \le i < \omega$) using the basic translation rules from Sect. 3.1. It is straightforward to check from the definition of the basic translation rule for the $\mathsf{R_w}$ connective (Table 3.1, p. 44) that

$$\begin{aligned}\Delta^I_{\varphi_2} & = \big\{\big\langle q_{\varphi_2}, \Gamma_{p_1}, \emptyset, \{q_{\varphi_2}\}\big\rangle, \langle q_{\varphi_2}, \Gamma_{p_2} \cap \Gamma_{p_1}, \emptyset, \emptyset\rangle\big\} \text{ holds, and} \\ \Delta^I_{\varphi_{i+1}} & = \big\{\big\langle q_{\varphi_{i+1}}, \Gamma, \emptyset, Q' \cup \{q_{\varphi_{i+1}}\}\big\rangle \,\big|\, \langle q_{\varphi_i}, \Gamma, F, Q'\rangle \in \Delta^I_{\varphi_i}\big\} \\ & \quad \cup \big\{\langle q_{\varphi_{i+1}}, \Gamma_{p_{i+1}} \cap \Gamma, \emptyset, Q'\rangle \,\big|\, \langle q_{\varphi_i}, \Gamma, F, Q'\rangle \in \Delta^I_{\varphi_i}\big\}\end{aligned}$$

holds for all $2 \leq i < \omega$. Denote the two components of $\Delta^I_{\varphi_{i+1}}$ in the above equation by $\Delta^{\text{sl}}_{\varphi_{i+1}}$ and $\Delta^{\text{nsl}}_{\varphi_{i+1}}$; obviously, $\Delta^{\text{sl}}_{\varphi_{i+1}}$ consists of the initial self-loops of the automaton built for the formula φ_{i+1}, and $\Delta^{\text{nsl}}_{\varphi_{i+1}}$ comprises those initial transitions of the automaton that are not self-loops. It is easy to check by induction on i that $|\Delta^I_{\varphi_i}| = 2^{i-1}$ holds for all $2 \leq i < \omega$, and $\Gamma \neq \emptyset$ holds for every guard Γ of a transition in $\Delta^I_{\varphi_i}$. Furthermore, for all $2 \leq i < j < \omega$, $\sigma \in 2^{AP}$ and $t = \langle q, \Gamma, F, Q' \rangle \in \Delta^I_{\varphi_i}$, $\sigma \in \Gamma$ holds only if both $\sigma \cup \{p_j\} \in \Gamma$ and $\sigma \setminus \{p_j\} \in \Gamma$ hold.

We claim that for all $2 \leq i < \omega$ and all pairs of transitions $t_1, t_2 \in \Delta^I_{\varphi_i}$ ($t_1 \neq t_2$), the transition t_2 is not substitutable for the transition t_1 in the sense of Gastin and Oddoux. Suppose that there exists a least index $2 \leq i < \omega$ such that $\Delta^I_{\varphi_i}$ contains a pair of transitions $t_1 = \langle q_{\varphi_i}, \Gamma_1, F_1, Q'_1 \rangle \in \Delta^I_{\varphi_i}$ and $t_2 = \langle q_{\varphi_i}, \Gamma_2, F_2, Q'_2 \rangle \in \Delta^I_{\varphi_i}$ ($t_1 \neq t_2$) such that the transition t_2 is substitutable for the transition t_1 in the sense of Gastin and Oddoux, i.e., $\Gamma_1 \subseteq \Gamma_2$ and $Q'_2 \subseteq Q'_1$ hold. Obviously, this cannot be the case if $i = 2$, because $\Gamma_{p_1} \not\subseteq \Gamma_{p_2} \cap \Gamma_{p_1}$ and $\{q_{\varphi_2}\} \not\subseteq \emptyset$. Therefore $i > 2$ holds. There are the following cases:

<u>($t_1 \in \Delta^{\text{sl}}_{\varphi_i}, t_2 \in \Delta^{\text{nsl}}_{\varphi_i}$)</u> By the definition of $\Delta^I_{\varphi_i}$, Γ_1 is the guard of a transition in $\Delta^I_{\varphi_{i-1}}$ in this case. Because $\Gamma_1 \neq \emptyset$ holds, $\sigma \in \Gamma_1$ holds for some $\sigma \in 2^{AP}$; then also $\sigma \setminus \{p_i\} \in \Gamma_1$ holds. But then $\sigma \notin \Gamma_2$ holds, because obviously $\sigma' \in \Gamma_2$ holds only if $p_i \in \sigma'$. Therefore $\Gamma_1 \not\subseteq \Gamma_2$, contrary to the assumption.

<u>($t_1 \in \Delta^{\text{nsl}}_{\varphi_i}, t_2 \in \Delta^{\text{sl}}_{\varphi_i}$)</u> In this case $q_{\varphi_i} \in Q'_2$ holds, because t_2 is a self-loop. Because t_1 is not a self-loop, however, $q_{\varphi_i} \notin Q'_1$ holds. But then it cannot be the case that $Q'_2 \subseteq Q'_1$ holds.

<u>($t_1, t_2 \in \Delta^{\text{sl}}_{\varphi_i}$)</u> By the definition of $\Delta^I_{\varphi_i}$, $\Delta^I_{\varphi_{i-1}}$ contains initial transitions $\hat{t}_1 = \langle q_{\varphi_{i-1}}, \Gamma_1, \widehat{F}_1, \widehat{Q}_1 \rangle \in \Delta^I_{\varphi_{i-1}}$ and $\hat{t}_2 = \langle q_{\varphi_{i-1}}, \Gamma_2, \widehat{F}_2, \widehat{Q}_2 \rangle \in \Delta^I_{\varphi_{i-1}}$ such that $\widehat{Q}_1 = Q'_1 \setminus \{q_{\varphi_i}\}$ and $\widehat{Q}_2 = Q'_2 \setminus \{q_{\varphi_i}\}$ hold; because $t_1 \neq t_2$ holds, $\hat{t}_1 \neq \hat{t}_2$ holds. Because $\Gamma_1 \subseteq \Gamma_2$ and $\widehat{Q}_2 = Q'_2 \setminus \{q_{\varphi_i}\} \subseteq Q'_1 \setminus \{q_{\varphi_i}\} = \widehat{Q}_1$ hold, the transition $\hat{t}_2$ is substitutable for $\hat{t}_1$ in the sense of Gastin and Oddoux. But then i is cannot the least index for which $\Delta^I_{\varphi_i}$ contains a pair of transitions, one of which is substitutable for the other in this sense.

<u>($t_1, t_2 \in \Delta^{\text{nsl}}_{\varphi_i}$)</u> In this case the automaton built for the formula φ_{i-1} has the initial transitions $\hat{t}_1 = \langle q_{\varphi_{i-1}}, \widehat{\Gamma}_1, \widehat{F}_1, Q'_1 \rangle \in \Delta^I_{\varphi_{i-1}}$ and $\hat{t}_2 = \langle q_{\varphi_{i-1}}, \widehat{\Gamma}_2, \widehat{F}_2, Q'_2 \rangle \in \Delta^I_{\varphi_{i-1}}$ such that $\Gamma_1 = \Gamma_{p_i} \cap \widehat{\Gamma}_1$ and $\Gamma_2 = \Gamma_{p_i} \cap \widehat{\Gamma}_2$ hold. Clearly, $\hat{t}_1 \neq \hat{t}_2$ holds, since otherwise also t_1 and t_2 would be identical transitions. Suppose that $\sigma \in \widehat{\Gamma}_1$ holds for some $\sigma \in 2^{AP}$. Then also $\sigma \cup \{p_i\} \in \widehat{\Gamma}_1$ holds, and thus $\sigma \cup \{p_i\} \in \Gamma_{p_i} \cap \widehat{\Gamma}_1 = \Gamma_1 \subseteq \Gamma_2 = \Gamma_{p_i} \cap \widehat{\Gamma}_2 \subseteq \widehat{\Gamma}_2$. But then also $\big(\sigma \cup \{p_i\}\big) \setminus \{p_i\} = \sigma \in \widehat{\Gamma}_2$. It follows that $\widehat{\Gamma}_1 \subseteq \widehat{\Gamma}_2$ holds, and because $Q'_2 \subseteq Q'_1$ holds, the transition $\hat{t}_2$ is substitutable for $\hat{t}_1$ in the sense of Gastin and Oddoux. Similarly to the previous case, i cannot be the least index for which $\Delta^I_{\varphi_i}$ contains two transitions, one of which can be substituted for the other.

Because all cases yield a contradiction, it follows that no transition in $\Delta^I_{\varphi_i}$ is substitutable for another transition in the sense of Gastin and Oddoux; by induction, the claim holds for all $2 \leq i < \omega$. Therefore, transition substitutability in the sense of Gastin and Oddoux does not help in removing initial transitions (by applying Proposition 6.2.2 or Proposition 6.2.3) from automata built for formulas in the family $\{\varphi_n\}_{2\leq n<\omega}$ using the basic translation rules, and it follows that the number of transitions in the automaton built for the formula φ_i is exponential in i. More precisely, it is straightforward to check that the subautomaton rooted at the initial state of the automaton built for the formula φ_i ($2 \leq i < \omega$) has $\sum_{j=2}^{i} |\Delta_j^I| = \sum_{j=2}^{i} 2^{j-1} = 2^i - 2 \in O(2^i)$ transitions (for all $3 \leq i < \omega$, $\Delta_{\varphi_i}^I$ contains a transition having the initial state of the automaton built from the formula φ_{i-1} as a target state).

(Translation using the refined rules) We now consider the translation of formulas in the family $\{\varphi_n\}_{2\leq n<\omega}$ into alternating automata using a refined translation rule for the $\mathsf{R_w}$ connective from Table 5.3 (p. 143). (We use the rule that does not involve assumptions on language containment; it can be shown that the other rule is never applicable when working with subformulas of formulas in the family.) As above, let q_{φ_i} and $\Delta_{\varphi_i}^I$ ($2 \leq i < \omega$) denote the initial state and the initial transitions of an automaton built for the formula φ_i. Obviously, the automaton built for the formula φ_2 is identical to the one built using the basic translation procedure, i.e.,

$$\Delta_{\varphi_2}^I = \big\{\big\langle q_{\varphi_2}, \Gamma_{p_1}, \emptyset, \{q_{\varphi_2}\}\big\rangle, \langle q_{\varphi_2}, \Gamma_{p_2} \cap \Gamma_{p_1}, \emptyset, \emptyset\rangle\big\}$$

holds also in this case.

Let $2 \leq i < \omega$. Assume that $\Delta_{\varphi_i}^I$ contains the self-loop $\big\langle q_{\varphi_i}, \Gamma_{p_1}, \emptyset, \{q_{\varphi_i}\}\big\rangle$, and $\Gamma \subseteq \Gamma_{p_1}$ holds for all guards Γ of transitions in $\Delta_{\varphi_i}^I$. (Clearly, these assumptions hold if $i = 2$.)

Let $\widetilde{\Delta}_{\varphi_{i+1}}^I$ be the set of initial transitions in the automaton built for the formula φ_{i+1} by applying the refined translation rule for the $\mathsf{R_w}$ connective to define $\widetilde{\Delta}_{\varphi_{i+1}}^I$ from $\Delta_{p_{i+1}}^I$ and $\Delta_{\varphi_i}^I$. From the definition of the rule (Table 5.3) it follows that

$$\begin{aligned}\widetilde{\Delta}_{\varphi_{i+1}}^I = {} & \big\{\big\langle q_{\varphi_{i+1}}, \Gamma, F, (Q' \setminus \{q_{\varphi_i}\}) \cup \{q_{\varphi_{i+1}}\}\big\rangle \;\big|\; \langle q_{\varphi_i}, \Gamma, F, Q'\rangle \in \Delta_{\varphi_i}^I\big\} \\ & \cup \big\{\langle q_{\varphi_{i+1}}, \Gamma_{p_{i+1}} \cap \Gamma, \emptyset, Q'\rangle \;\big|\; \langle q_{\varphi_i}, \Gamma, F, Q'\rangle \in \Delta_{\varphi_i}^I\big\}.\end{aligned}$$

It now follows from the assumptions that $\widetilde{\Delta}_{\varphi_{i+1}}^I$ contains the self-loop $t = \big\langle q_{\varphi_{i+1}}, \Gamma_{p_1}, \emptyset, \{q_{\varphi_{i+1}}\}\big\rangle$, and $\Gamma \subseteq \Gamma_{p_1}$ holds also for all guards Γ of transitions in $\widetilde{\Delta}_{\varphi_{i+1}}^I$. Furthermore, the self-loop t is substitutable for all self-loops in $\widetilde{\Delta}_{\varphi_{i+1}}^I \setminus \{t\}$ in the sense of Gastin and Oddoux, since all self-loops in $\widetilde{\Delta}_{\varphi_{i+1}}^I$ are of the form $\big\langle q_{\varphi_{i+1}}, \Gamma, F, (Q' \setminus \{q_{\varphi_i}\}) \cup \{q_{\varphi_{i+1}}\}\big\rangle$ for some transition $\langle q_{\varphi_i}, \Gamma, F, Q'\rangle \in \Delta_{\varphi_i}^I$: by the above observation, $\Gamma \subseteq \Gamma_{p_1}$ holds, and obviously also $\{q_{\varphi_{i+1}}\} \subseteq (Q' \setminus \{q_{\varphi_i}\}) \cup \{q_{\varphi_{i+1}}\}$ holds. Because the set of acceptance conditions of the transition t is empty, it follows by Proposition 6.2.3 that all self-loops in $\widetilde{\Delta}_{\varphi_{i+1}}^I \setminus \{t\}$ are redundant. We may thus write the set of initial transitions in the automaton built for the formula φ_{i+1} (after removing redundant transitions) as

$$\begin{aligned}\Delta_{\varphi_{i+1}}^I = {} & \big\{\big\langle q_{\varphi_{i+1}}, \Gamma_{p_1}, \emptyset, \{q_{\varphi_{i+1}}\}\big\rangle\big\} \\ & \cup \big\{\langle q_{\varphi_{i+1}}, \Gamma_{p_{i+1}} \cap \Gamma, \emptyset, Q'\rangle \;\big|\; \langle q_{\varphi_i}, \Gamma, F, Q'\rangle \in \Delta_{\varphi_i}^I\big\}.\end{aligned}$$

Because $\Delta^I_{\varphi_{i+1}}$ contains the self-loop $\langle q_{\varphi_{i+1}}, \Gamma_{p_1}, \emptyset, \{q_{\varphi_{i+1}}\}\rangle$, and because $\Gamma \subseteq \Gamma_{p_1}$ holds for all guards Γ of transitions in $\widetilde{\Delta}^I_{\varphi_{i+1}}$, the above equality holds for all $2 \leq i < \omega$ by induction on i.

It is straightforward to check (using similar arguments as for the basic translation rules) that no transition in $\Delta^I_{\varphi_{i+1}}$ is substitutable for another transition in $\Delta^I_{\varphi_{i+1}}$ in the sense of Gastin and Oddoux. Because no two distinct transitions in $\Delta^I_{\varphi_i}$ map into the same transition in $\Delta^I_{\varphi_{i+1}}$ in the definition of $\Delta^I_{\varphi_{i+1}}$, it follows that $|\Delta^I_i| = i$ holds in this case for all $2 \leq i < \omega$ (clearly, $|\Delta^I_{\varphi_2}| = 2$ holds, and if $|\Delta^I_{\varphi_i}| = i$ holds for some $2 \leq i < \omega$, then obviously $|\Delta^I_{\varphi_{i+1}}| = i+1$). It is easy to check that the subautomaton rooted at the initial state of the automaton built in this way for the formula φ_i $(2 \leq i < \omega)$ then has $\sum_{j=2}^{i} |\Delta^I_j| = \sum_{j=2}^{i} j = \frac{i\cdot(i+1)}{2} - 1 \in O(i^2)$ transitions (again, it is easy to check that $\Delta^I_{\varphi_i}$ contains a transition having the initial state of the automaton built from the formula φ_{i-1} as a target state for all $3 \leq i < \omega$). As a matter of fact, it is actually the case that the automaton built for the formula φ_i has no transitions with two or more target states: thus every formula in the family $\{\varphi_n\}_{2\leq n<\omega}$ can be translated into a self-loop nondeterministic automaton with a linear number of states in the length of the formula. ■

Not All Redundant Initial Transitions Can Be Detected

All previous examples in this section were based on using Proposition 6.2.2 and Proposition 6.2.3 to detect redundant initial transitions in self-loop alternating automata. It is nevertheless easy to show that these results are not powerful enough for finding all redundant initial transitions in self-loop alternating automata.

Example 6.2.14 Figure 6.4 (a) depicts the self-loop alternating automaton built from the LTL formula $\big((\bot\, \mathsf{R_w}\, (p_1 \wedge p_2))\, \mathsf{R_w}\, p_1\big)$ using the refined translation rules. It is easy to see that this automaton fin-accepts a word $w \in (2^{\{p_1,p_2\}})^\omega$ iff $p_1 \in w(i)$ holds for all $0 \leq i < \omega$. Therefore the automaton is fin-equivalent to the automaton obtained from it by removing its initial non-self-loop transition (Fig. 6.4 (b)). Hence, this transition is redundant; however, neither Proposition 6.2.2 nor Proposition 6.2.3 applies in this case (the only transition that can be substituted for the transition under fin-acceptance is a self-loop, but the substituted transition itself is not a self-loop). ■

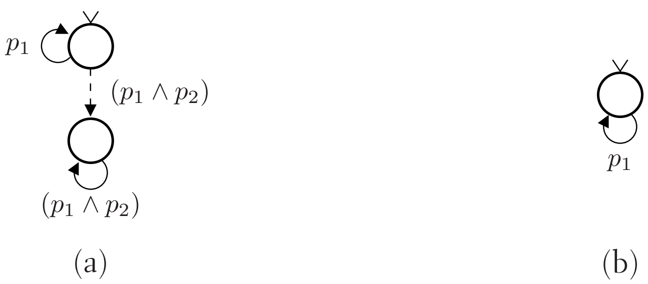


Fig. 6.4: Not all redundant initial transitions of self-loop alternating automata can be detected by applying Proposition 6.2.2 or Proposition 6.2.3. (a) Self-loop alternating automaton built from the formula $\big((\bot \;\mathsf{R_w}\; (p_1 \wedge p_2)) \;\mathsf{R_w}\; p_1\big)$; (b) Automaton obtained from (a) by removing a redundant initial transition that cannot be detected using one of the propositions

7 A HIGH-LEVEL REFINED TRANSLATION PROCEDURE

In this short chapter we summarize how the results presented in the two previous chapters can be combined with the translation procedure from LTL into self-loop alternating automata.

Let $\varphi \in LTL(AP)$ be an LTL formula over the atomic propositions AP. The translation of the formula φ into a self-loop alternating automaton starts by rewriting φ in positive normal form (optionally, applying syntactic optimizations to the formula as is common in translation algorithms [Etessami and Holzmann 2000; Somenzi and Bloem 2000; Thirioux 2002]). Let φ' be the formula (in positive normal form) obtained from this rewriting step. A self-loop alternating automaton for the formula φ' can then be defined by repeating the following high-level steps:

1. Choose a formula $\psi \in \mathrm{NSub}(\varphi')$ that has not yet been translated into an automaton such that automata for all top-level subformulas of ψ have already been constructed. If ψ is an atomic formula, build a single-state automaton for ψ using the translation rule for the atomic formulas (Sect. 3.1) and repeat from this step.

2. Check whether any of the identities discussed in Sect. 5.3 apply to ψ by analyzing the automata built for the top-level subformulas of ψ; if ψ reduces to a formula for which an automaton already exists, reuse that automaton for ψ and return to step 1.

3. Choose a translation rule according to the main connective of ψ and the relationship between the top-level subformulas of ψ (Table 3.1, Table 5.2, Table 5.3) and apply it to the automata built for the top-level subformula(s) of ψ to obtain an automaton $\mathcal{A}_\psi$.

4. Remove redundant initial transitions from $\mathcal{A}_\psi$ as discussed in Sect. 5.1 and Sect. 6.1 (by applying Corollary 2.3.11 and the results proved in Proposition 6.2.2, Proposition 6.2.3 and Sect. 6.2.5).

5. Repeat from step 1 until $\mathcal{A}_{\varphi'}$ has been constructed.

In principle, the above procedure can easily be augmented with additional minimization tests, for example, to reduce $\mathcal{A}_\psi$ into a single-state automaton corresponding to one of the Boolean constants if it accepts the empty or the universal language. Furthermore, instead of only removing redundant transitions from the automaton, it is possible to use language containment tests also for replacing states systematically with their language-equivalent representatives in each remaining transition or to reduce the number of target states in the transitions [Rohde 1997].

As observed in Sect. 5.2 and Sect. 6.2.4, every language containment test applied in the above procedure can be reduced to testing the emptiness of the set intersection of languages recognized by self-loop alternating automata built from subformulas of the formula φ' or the positive normal forms of their negations. Because the formula φ' is known at the beginning of the translation, these automata can be found directly by recursive invocations of the above translation procedure. The number of these invocations is obviously bounded by $2 \cdot |\mathrm{NSub}(\varphi')|$ if no automaton built in the translation is discarded until the end of the translation. The collection of the self-loop alternating automata built in the translation can then be seen as a single self-loop alternating automaton $\mathcal{A}$ having at most $2 \cdot |\mathrm{NSub}(\varphi')|$ states corresponding to formulas in the set $\mathrm{NSub}(\varphi') \cup \mathrm{NSub}\big([\neg\varphi']^{\mathrm{PNF}}\big)$. Every collection of alternating automata referred to in a test for language emptiness can then be seen as a collection of subautomata of the underlying automaton $\mathcal{A}$.

To effectively check whether the set intersection of languages recognized by a collection of subautomata of $\mathcal{A}$ (determined by a set Q of their initial states) is empty, it is now straightforward to apply a language emptiness checking procedure for nondeterministic automata to the subautomaton rooted at the state Q of the nondeterministic automaton $\mathcal{A}'$ built from $\mathcal{A}$ by applying the universal subset construction, since $\mathcal{L}_{\mathrm{fin}}\big((\mathcal{A}')^Q\big) = \bigcap_{q \in Q} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$ holds by Proposition 4.4.1 (Sect. 4.4). (Because all translation rules and simplification techniques used in the above procedure generate acceptance closed self-loop alternating automata with representative states for all of their acceptance conditions, the automaton $\mathcal{A}$ can be nondeterminized without introducing new acceptance conditions as observed in Corollary 4.3.7; also the optimizations presented in Sect. 4.5 apply to the nondeterminization procedure.) As usual, the universal subset construction can be embedded directly into a language emptiness checking algorithm for nondeterministic automata which is then invoked incrementally on subautomata of the automaton $\mathcal{A}'$ during the translation procedure. In particular, language emptiness checking algorithms based on Tarjan's algorithm for finding the strongly connected components of a directed graph [Tarjan 1971, 1972] are easily adaptable to compute the answer to the language emptiness question for the automaton $(\mathcal{A}')^Q$ together with all of its subautomata in the same pass of the state space of the automaton $(\mathcal{A}')^Q$. (Of the emptiness checking algorithms based on finding the strongly connected components of a nondeterministic automaton [Couvreur 1999; Geldenhuys and Valmari 2004, 2005; Hammer et al. 2005; Schwoon and Esparza 2005; Couvreur et al. 2005], the algorithm of Couvreur [1999]—see also [Couvreur et al. 2005]—is perhaps best compatible with our definition of automata since it applies directly to automata with generalized transition-based acceptance.) Again, if no subautomaton of the nondeterministic automaton is discarded in the translation, all language emptiness checks can thus be handled (in effect) in a single invocation of a language emptiness checking algorithm on the nondeterministic automaton $\mathcal{A}'$ (which has at most $2^{2\cdot|\mathrm{NSub}(\varphi')|}$ states). Technically, even though the structure of the underlying alternating automaton $\mathcal{A}$ may change between invocations of an emptiness checking procedure (for example, if some transitions are removed from the automaton), it is nevertheless not necessary to repeat a language emptiness check for any collection of subautomata of $\mathcal{A}$ because all of our simplifications preserve the language of each subautomaton of $\mathcal{A}$.

The language containment based optimizations used in the high-level translation procedure do not affect the order of the upper bound for the number of recursive invocations of the basic translation procedure, or the number of states in a nondeterministic automaton built in the translation of an LTL formula into a nondeterministic automaton via a self-loop alternating automaton (these upper bounds remain of the order $O\big(|\mathrm{NSub}(\varphi')|\big)$ and $2^{O(|\mathrm{NSub}(\varphi')|)}$, respectively). Nevertheless, it is to be expected that this translation procedure is realistically applicable in practice only to small problem instances. Because the translation procedure enhanced with language containment based optimizations is intuitively more prone to perform the worst-case amount of work than the basic translation procedure, the performance penalty resulting from repeated unsuccessful language containment tests is likely to become clearly visible in practice (in favor of a procedure without language containment based optimizations) in cases where the language containment tests fail to discover opportunities for simplification.

By the design of our language containment based optimizations, every optimization depends on a positive answer to a question on language containment (equivalently, language emptiness) in order to be applicable. Therefore it is always safe to assume the answer to such a question to be negative (consequently, disabling the optimization) without risking the correctness of the translation procedure. For example, the effort spent in recursive invocations of the translation procedure for formulas that do not occur as node subformulas of φ' could be reduced by specifying a bound for the node size (or other syntactic measure, such as the number of temporal operators) of LTL formulas for which to invoke the translation procedure recursively. Similarly, the amount of work needed for building nondeterministic automata for language emptiness tests could be controlled by limiting the maximum number of states of the underlying nondeterministic automaton to generate in the translation, disabling the language emptiness tests if this bound is exceeded during the translation. Instead of applying the universal subset construction, the language emptiness checks could also be approximated by syntactic techniques (Sect. 4.5.1) to try to detect the unsatisfiability of a conjunction of LTL formulas corresponding to a set of alternating automata. Additionally, caching the results of the emptiness checks for subsets of the states of the alternating automaton $\mathcal{A}$ would allow the explicit construction of a nondeterministic automaton to be avoided in some cases for other subsets of states of the automaton: for example, if $\bigcap_{q\in Q}\mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)=\emptyset$ is known to hold for a subset Q of states of $\mathcal{A}$, then obviously $\bigcap_{q\in Q'}\mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)=\emptyset$ holds also for all sets Q' of which Q is a subset; conversely, if $\bigcap_{q\in Q}\mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)\neq\emptyset$ holds, then $\bigcap_{q\in Q'}\mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)\neq\emptyset$ holds for all subsets $Q'\subseteq Q$.

8 LANGUAGE EMPTINESS CHECKING FOR NONDETERMINISTIC AUTOMATA

Checking for the emptiness of the language recognized by a nondeterministic automaton working in inf- or fin-acceptance mode can be reduced to a question on repeated reachability in automata (see, for example, [Vardi and Wolper 1994]). Therefore, the emptiness checking problem can be solved by using basic graph algorithms to decide the existence of cycles satisfying certain acceptance constraints. In particular, Tarjan's algorithm for finding the maximal strongly connected components of a directed graph in linear time in the number of nodes and edges in the graph using depth-first search [Tarjan 1971, 1972] has been found to be easily and efficiently adaptable to language emptiness checking for nondeterministic automata working in inf- or fin-acceptance mode using one [Geldenhuys and Valmari 2004; Schwoon and Esparza 2005] or more [Couvreur 1999; Hammer et al. 2005; Couvreur et al. 2005; Geldenhuys and Valmari 2005] acceptance conditions. Because the algorithms can be combined with the on-the-fly construction of a nondeterministic automaton, they are often able to detect language nonemptiness without constructing the automaton in full. This on-the-fly detection of language nonemptiness is useful especially in model checking applications, where the nondeterministic automata obtained by composing system descriptions with automata expressing correctness specifications (built, for example, from LTL formulas) are often very large due to state space explosion.

The single most critical resource consumed by an emptiness checking algorithm based on a search in the state space of an automaton is usually considered to be the amount of randomly accessed memory needed for distinguishing already visited states from states that have not been explored in the search, together with the additional memory used for implementing the actual language emptiness check on top of the search. For example, variants of Tarjan's algorithm commonly number the states of the automaton in the order in which they are encountered during the search, associating each state with one or more integers whose maximum possible values depend on the number of states in the automaton. Even though some variants of language emptiness checking algorithms try to discard as much information as possible when it is not needed any longer in the search by maximizing the use of stacks in place of hash tables for storing the information [Geldenhuys and Valmari 2004, 2005], the worst-case memory requirements of these algorithms still exceed those of the *nested depth-first search* (NDFS) algorithm of Courcoubetis et al. [1991, 1992] and its many variants [Godefroid and Holzmann 1993; Holzmann et al. 1997; Edelkamp et al. 2001, 2004; Brim et al. 2001; Bošnački 2002, 2003; Gastin et al. 2004; Tauriainen 2004, 2005, 2006; Schwoon and Esparza 2005; Couvreur et al. 2005]. Instead of analyzing the strongly connected components of an automaton, these algorithms try to detect the nonemptiness of the language of the automaton via the reduction of language nonemptiness to repeated reachability. However, the worst-case number of passes of the state space of the automaton required by these algorithms depends on the number of acceptance conditions in the automaton. It is therefore not surprising that variants of Tarjan's algorithm, for which the number of passes is independent of the number of acceptance conditions, outperform NDFS-based ones as regards the number of states (re-)explored in the search [Schwoon and Esparza 2005; Couvreur et al. 2005].

In addition to using multiple passes of the state space, most known variants of the basic nested depth-first search algorithm are not directly applicable to automata with more than one acceptance condition: the automata first need to be *degeneralized* to replace the multiple acceptance conditions with a single one [Emerson and Sistla 1984a,b; Courcoubetis et al. 1991, 1992; Gastin and Oddoux 2001; Giannakopoulou and Lerda 2002]. Although a simple operation in a theoretical sense, degeneralization may nevertheless increase the number of states in the automaton by a factor that is proportional to the number of acceptance conditions. Because the increase in the number of states cannot be avoided in the general case, the transformation may thus work against any memory savings gained from the use of a special emptiness checking algorithm. Other advantages of direct emptiness checking algorithms for automata with multiple acceptance conditions have previously been pointed out in the context of verification under simple liveness [Aggarwal et al. 1990] (or weak fairness) assumptions by Couvreur [1999], who proposed an on-the-fly variant of Tarjan's algorithm [Tarjan 1971, 1972] for checking the emptiness of languages recognized by automata with multiple acceptance conditions, and Bošnački [2003], who studied combining the nested depth-first search with fairness and symmetry reduction.

In this chapter, we present a variant of the basic nested depth-first search emptiness checking algorithm of [Courcoubetis et al. 1991, 1992], combining material from previous work by the author in [Tauriainen 2004, 2005, 2006]. This variant is designed to work directly on automata with multiple acceptance conditions and thus avoids the need for degeneralization. Instead of using automata working in fin-acceptance mode, we switch our focus to automata working in inf-acceptance mode to make our algorithm directly substitutable for other constructions known from the literature. When dealing with nondeterministic automata obtained from self-loop alternating automata working in fin-acceptance mode, the simple correspondence between the acceptance modes allows the inversion of the acceptance mode to be combined, for example, with the nondeterminization of self-loop alternating automata (which can by itself be embedded into the language emptiness checking algorithm [Hammer et al. 2005]).

Although the algorithm presented in this chapter could be used to implement language emptiness checks in the high-level translation procedure for translating LTL formulas into self-loop alternating automata (Ch. 7), the algorithm is geared more towards model checking (variants of Tarjan's algorithm work faster, and they are easily adaptable to answer the language emptiness question for all subautomata of a nondeterministic automaton in the same pass of the automaton's state space).

8.1 TERMINOLOGY

In this section we briefly introduce additional terminology which we shall use in the analysis of nondeterministic automata working in inf-acceptance mode.

Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a nondeterministic automaton, and let $(t_i)_{0 \le i < n} \in \Delta^n$ (for some $0 \le n \le \omega$) be a chain of transitions in $\mathcal{A}$. We say that the chain $(t_i)_{0 \le i < n}$ *fulfills* the acceptance condition $f \in \mathcal{F}$ iff there exists an index $0 \le i < n$ such that the transition t_i is an f-transition of $\mathcal{A}$.

Because every transition of $\mathcal{A}$ has exactly one target state, it is easy to see that every nonempty chain of transitions in $\mathcal{A}$ determines a unique path in $\mathcal{A}$. Thus we say that a nonempty chain of transitions *visits* a state $q \in Q$ if the path corresponding to it does; similarly, we may consider a state $q' \in Q$ to be reachable from another state $q \in Q$ via a nonempty chain of transitions if q' is reachable from q via the path that corresponds to the chain of transitions, and we may call a finite nonempty chain of transitions a *cycle* if the path determined by the chain is a cycle of $\mathcal{A}$.

It is easy to see that every infinite chain of transitions with nonempty guards that begins with an initial transition of $\mathcal{A}$ defines a run of $\mathcal{A}$ on some input and vice versa. Therefore, because Δ is finite, it follows that $\mathcal{A}$ has an inf-accepting run on some input $w \in \Sigma^\omega$ (that is, $\mathcal{L}_{\text{inf}}(\mathcal{A}) \neq \emptyset$ holds) iff $\mathcal{A}$ contains an *accepting cycle*, i.e., a cycle (of transitions with nonempty guards) which fulfills all acceptance conditions $f \in \mathcal{F}$ and visits a state that is reachable from the initial state q_I of the automaton.

Finally, a *maximal strongly connected component* of $\mathcal{A}$ is a maximal subset $C \subseteq Q \cup \Delta$ such that for all $q, q' \in C \cap Q$, there exists a path from q to q' in $\mathcal{A}$ that corresponds to a (possibly empty) chain of transitions with nonempty guards, and for all $t = \langle q, \Gamma, F, \{q'\} \rangle \in C \cap \Delta$, $\Gamma \neq \emptyset$ and $q, q' \in C$ hold. It is straightforward to check that every accepting cycle of $\mathcal{A}$ is contained in a maximal strongly connected component of $\mathcal{A}$ that contains an f-transition for all $f \in \mathcal{F}$.

8.2 DEGENERALIZATION

It is well-known that nondeterministic automata with more than one acceptance condition are not more expressive than automata with a single acceptance condition, and there are effective *degeneralization* constructions for replacing the acceptance conditions in a nondeterministic automaton with a single condition. These constructions [Emerson and Sistla 1984a,b; Courcoubetis et al. 1991, 1992] can be seen either as variants of the "flag-construction" of Choueka [1974] for defining a nondeterministic ω-automaton for the set intersection of the languages recognized by a collection of nondeterministic ω-automata, or as procedures that achieve their goal by combining automata with special "degeneralizer" automata [Couvreur 1999; Giannakopoulou and Lerda 2002].

It is straightforward to adapt, for example, the construction proposed by Courcoubetis et al. [1991, 1992] (designed for automata with state-based acceptance) to automata with transition-based acceptance. Intuitively, the construction works by taking as many copies of the automaton as there are acceptance conditions, and then connecting the copies together into an automaton in which every finite chain of transitions that begins and ends with transitions fulfilling the acceptance condition of the automaton corresponds to a chain that fulfills all acceptance conditions in the original automaton (and vice versa). Therefore, any infinite chain of transitions satisfying the inf-acceptance condition in one of the automata corresponds to a similar chain in the other automaton.

Formally, the construction transforms a nondeterministic automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ with state set $Q = \{q_1, q_2, \ldots, q_n\}$ (where $1 \leq n < \omega$ and $q_I = q_1$) and a nonempty set of acceptance conditions $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ ($1 \leq m < \omega$) into a nondeterministic automaton $\mathcal{A}' = \big\langle \Sigma, Q', \Delta', q'_I, \{f\} \big\rangle$, where

$$
\begin{aligned}
Q' &\overset{\mathit{def}}{=} \{q_{(i,j)}\}_{i=1,j=1}^{i=n,j=m}, \\
\Delta' &\overset{\mathit{def}}{=} \bigcup_{k=1}^{m} \Big(\big\{ \big\langle q_{(i,k)}, \Gamma, \emptyset, \{q_{(j,k)}\} \big\rangle \,\big|\, \big\langle q_i, \Gamma, F, \{q_j\} \big\rangle \in \Delta,\, f_k \notin F \big\} \\
&\qquad\quad \cup \big\{ \big\langle q_{(i,k)}, \Gamma, F(i), \{q_{(j,1+(k \bmod m))}\} \big\rangle \,\big|\, \big\langle q_i, \Gamma, F, \{q_j\} \big\rangle \in \Delta, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad f_k \in F \big\} \Big), \\
q'_I &\overset{\mathit{def}}{=} q_{(1,k)}, \quad \text{and} \\
F(i) &\overset{\mathit{def}}{=} \begin{cases} \{f\} & \text{if } i = \ell \text{ holds for some fixed } 1 \leq \ell \leq m \\ \emptyset & \text{otherwise.} \end{cases}
\end{aligned}
$$

(If $\mathcal{F} = \emptyset$ holds, we can define $\mathcal{A}' \overset{\mathit{def}}{=} \big\langle \Sigma, Q, \Delta', q_I, \{f\} \big\rangle$, where $\Delta' \overset{\mathit{def}}{=} \big\{ \big\langle q, \Gamma, \{f\}, \{q'\} \big\rangle \,\big|\, \big\langle q, \Gamma, \emptyset, \{q'\} \big\rangle \in \Delta \big\}$.)

When applied to an automaton $\mathcal{A}$ with n states and $1 \leq m < \omega$ acceptance conditions, the transformation yields an automaton $\mathcal{A}'$ with nm states and one acceptance condition such that the automata $\mathcal{A}'$ and $\mathcal{A}$ inf-recognize the same language [Emerson and Sistla 1984a,b; Courcoubetis et al. 1991, 1992]. The construction gives an $O(nm)$ worst-case lower bound for the number of states in an automaton obtained via degeneralization. As shown below, this lower bound is optimal, i.e., the linear blow-up in the number of states in the automaton cannot be avoided in the general case. (Although this result is intuitively very obvious, the optimality of the construction is usually not considered in literature on degeneralization algorithms [Emerson and Sistla 1984a,b; Courcoubetis et al. 1991, 1992; Clarke et al. 1999] or their optimizations [Gastin and Oddoux 2001; Giannakopoulou and Lerda 2002]. We therefore repeat an example from [Tauriainen 2004, 2006] here for illustration.)

Proposition 8.2.1 *A nondeterministic automaton built by degeneralizing a nondeterministic automaton with $1 \leq n < \omega$ states and $1 \leq m < \omega$ acceptance conditions (and an alphabet of at least $m+1$ symbols) needs nm states to recognize the language of the original automaton in the worst case.*

Proof: Let $2 \leq n < \omega$ be an integer, and let Σ_m denote a finite alphabet with $1 \leq m < \omega$ distinct symbols $\sigma_1, \ldots, \sigma_m$ together with an extra symbol $\#$ different from each σ_i. The parameters n and m can be used to define a set of infinite words $\mathcal{L}_{n,m}$ over Σ_m characterized by the ω-regular expression

$$
\begin{aligned}
\mathcal{L}_{n,m} = \Big(&\big((\textstyle\bigcup_{\substack{i=1\\ i\neq 1}}^{m} \sigma_i) \#^{n-1} \big)^{*} \sigma_1 \#^{n-1} \\
&\big((\textstyle\bigcup_{\substack{i=1\\ i\neq 2}}^{m} \sigma_i) \#^{n-1} \big)^{*} \sigma_2 \#^{n-1} \\
&\quad\cdots \\
&\big((\textstyle\bigcup_{\substack{i=1\\ i\neq m}}^{m} \sigma_i) \#^{n-1} \big)^{*} \sigma_m \#^{n-1} \Big)^{\omega},
\end{aligned}
$$

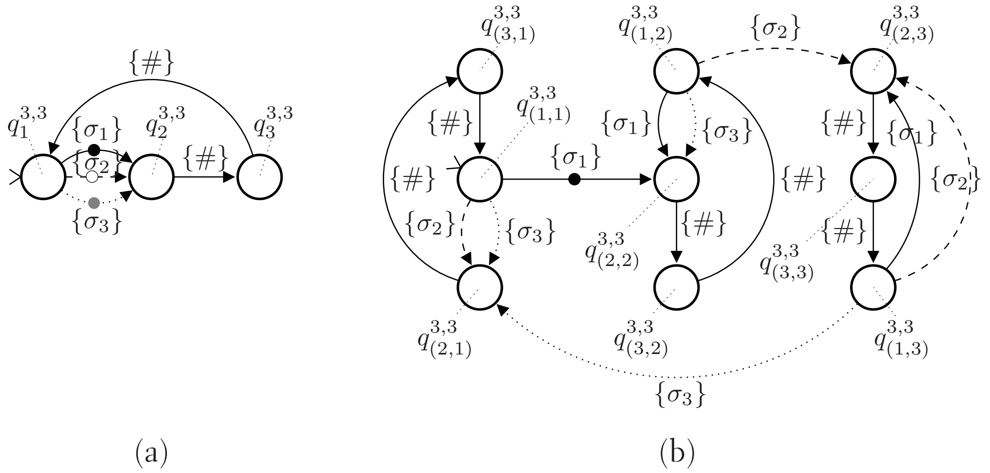


Fig. 8.1: Examples of automata used in the proof of Proposition 8.2.1. (a) The automaton $\mathcal{A}_{3,3}$. (b) Automaton obtained from $\mathcal{A}_{3,3}$ via degeneralization

i.e., the set of infinite words built from n-symbol "blocks" over Σ_m, each of which consists of one of the symbols σ_i ($1 \leq i \leq m$) followed by exactly $n-1$ $\#$'s, with the additional constraint that each σ_i should occur in the resulting word infinitely many times.

The language $\mathcal{L}_{n,m}$ can be recognized by the n-state nondeterministic automaton $\mathcal{A}_{n,m} = \langle \Sigma_m, Q_{n,m}, \Delta_{n,m}, q_I^{n,m}, \mathcal{F}_{n,m} \rangle$, where

$$\begin{aligned}
Q_{n,m} &\stackrel{\text{def}}{=} \{q_1^{n,m}, q_2^{n,m}, q_3^{n,m}, \ldots, q_n^{n,m}\}, \\
\Delta_{n,m} &\stackrel{\text{def}}{=} \left(\textstyle\bigcup_{i=1}^{m} \left\{\left\langle q_1^{n,m}, \{\sigma_i\}, \{f_m^n\}, \{q_2^{n,m}\}\right\rangle\right\}\right) \\
&\qquad\qquad \cup \left(\textstyle\bigcup_{i=2}^{n} \left\{\left\langle q_i^{n,m}, \{\#\}, \emptyset, \{q_{(i \bmod n)+1}^{n,m}\}\right\rangle\right\}\right), \\
q_I^{n,m} &\stackrel{\text{def}}{=} q_1^{n,m}, \quad \text{and} \\
\mathcal{F}_{n,m} &\stackrel{\text{def}}{=} \{f_1^n, f_2^n, \ldots, f_m^n\}.
\end{aligned}$$

As a concrete example, see Fig. 8.1 (a) for an illustration of the automaton $\mathcal{A}_{3,3}$.

By the above construction, there exists an automaton with nm states and a single acceptance condition such that the automaton recognizes the language $\mathcal{L}_{n,m}$. Figure 8.1 (b) shows the result when the construction is applied to the automaton $\mathcal{A}_{3,3}$. We argue that the construction is optimal by showing that any automaton that inf-recognizes the language $\mathcal{L}_{n,m}$ using only one acceptance condition always has at least nm states.

Let $\mathcal{A} = \left\langle \Sigma_m, Q, \Delta, q_I, \{f\} \right\rangle$ be an automaton that inf-recognizes $\mathcal{L}_{n,m}$ using only one acceptance condition f, and let $w \in \mathcal{L}_{n,m}$. Thus, $\mathcal{A}$ inf-accepts w, and there exists a uniform inf-accepting run G of $\mathcal{A}$ on w that consists of a unique infinite branch $\beta \in \mathcal{B}(G)$ such that $\inf(\beta) = \{f\}$ holds. Because Δ is finite, β contains infinitely many edges labeled with a transition $t = \left\langle q, \Gamma, F, \{q'\} \right\rangle \in \Delta$ such that $f \in F$ holds. Let u be a finite prefix of w such that the edge beginning at level $|u|$ of G is labeled with the transition t (with source state q). Because t occurs infinitely many times in G, there exists a chain of transitions (with nonempty guards) beginning with the transition t that corresponds to a cycle from q to itself in $\mathcal{A}$.

Consider any shortest chain of transitions (with nonempty guards) that begins with the transition t and corresponds to a cycle from q to itself in the automaton $\mathcal{A}$. Let v be a word composed of symbols in the guards of the successive transitions in the chain. It is easy to see that $\mathcal{A}$ inf-accepts the word uv^ω. Because $\mathcal{A}$ inf-recognizes the language $\mathcal{L}_{n,m}$, it follows that $uv^\omega \in \mathcal{L}_{n,m}$ holds. Hence each σ_i must occur at least once in v for all $1 \leq i \leq m$. In the simplest case, each σ_i occurs in v exactly once, and v is of the form $\#^k\sigma_{\rho(1)}\#^{n-1}\sigma_{\rho(2)}\#^{n-1}\cdots\sigma_{\rho(m)}\#^{n-k-1}$ for some $0 \leq k \leq n-1$ and some permutation ρ of $\{1,\ldots,m\}$. Clearly, v has $(n-1)m + m = nm$ symbols. Because v was formed from a shortest chain of transitions (beginning with the transition t) that corresponds to a cycle from q to itself, the source states of the transitions in the chain are distinct. It follows that $\mathcal{A}$ has at least nm states as argued. $\square$

8.3 EMPTINESS CHECKING ALGORITHM

Let $\mathcal{A} = \langle\Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ be a nondeterministic automaton with a nonempty set of acceptance conditions $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ for some $1 \leq n < \omega$. The indices of the acceptance conditions induce an ordering between the conditions. Figure 8.2 represents the pseudocode of an algorithm that decides whether $\mathcal{A}$ inf-recognizes the empty language or not by checking for the existence of an accepting cycle in $\mathcal{A}$. We present the algorithm in recursive form, from which it is easy to point out the relevant details. (If $\mathcal{F} = \emptyset$ holds, the emptiness of $\mathcal{L}_{\text{inf}}(\mathcal{A})$ can be decided using a single depth-first search in the automaton, because $\mathcal{L}_{\text{inf}}(\mathcal{A}) \neq \emptyset$ holds in this case iff $\mathcal{A}$ contains a cycle that visits a state reachable from the initial state q_I of $\mathcal{A}$. As a matter of fact, a single search is sufficient also for *weak* nondeterministic automata [Schwoon and Esparza 2005].)

The TOP-LEVEL-SEARCH procedure implements a simple depth-first search in the automaton. After processing a transition (line 7), a second search is started from the transition at line 9 (in further discussion, the term "second search" refers to all operations caused by a call to the SECOND-SEARCH procedure made at line 9). The purpose of this search is to propagate information about the reachability of states via chains of transitions which fulfill certain acceptance conditions. Each second search is restricted to states that have been found in the top-level search.

The algorithm uses the following (global) data structures:

path_stack: This stack collects the states entered in the top-level search but from which the search has not yet backtracked. This stack is used only to facilitate the correctness proof in Sect. 8.3.2 and thus all references to it (lines 3 and 12) could be removed from the pseudocode.

visited: States found in the top-level search. When a state is added to this set, it will never be removed.

count: A lookup table associating each state of the automaton with an index of an acceptance condition. Intuitively, if $\mathit{count}[q] = c$ holds for some $q \in Q$ and $1 \leq c \leq n$, then, for every acceptance condition $f_1, f_2, \ldots, f_c \in \mathcal{F}$, the automaton contains a nonempty chain of transitions (with nonempty guards) that corresponds to a path ending in

Initialize: $\mathcal{A} := \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$: Nondeterministic automaton with states $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ and $1 \le n < \omega$ acceptance conditions $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$;
$path\_stack$:= [empty stack];
$visited := \emptyset$;
$count := [q_1 \mapsto 0, \ldots, q_{|Q|} \mapsto 0]$;
$depth := 1$;
$condition\_stack$:= [stack initialized with the element $(0, 0)$].

```
TOP-LEVEL-SEARCH($q \in Q$)
  begin
    push $q$ on $path\_stack$;
    $visited := visited \cup \{q\}$;
    for all $t = \langle q, \Gamma, F, \{q'\} \rangle \in \Delta$ with $\Gamma \neq \emptyset$ do
      begin
        if ($q' \notin visited$) then TOP-LEVEL-SEARCH($q'$);
        $\mathcal{I}_{\mathrm{seen}} := \{1, 2, \ldots, count[q]\}$;
        SECOND-SEARCH($t$);
        if ($count[q] = |\mathcal{F}|$) then report "$\mathcal{L}_{\mathrm{inf}}(\mathcal{A}) \neq \emptyset$";
      end;
    pop $q$ off $path\_stack$;
  end

SECOND-SEARCH($\langle q, \Gamma, F, \{q'\} \rangle \in \Delta$)
  begin
    $\mathcal{I}_{\mathrm{unseen\_fulfilled}} := \{1 \le i \le |\mathcal{F}| \mid f_i \in F\} \setminus \mathcal{I}_{\mathrm{seen}}$;
    $c := \max\{0 \le i \le |\mathcal{F}| \mid j \in \mathcal{I}_{\mathrm{seen}} \cup \mathcal{I}_{\mathrm{unseen\_fulfilled}}$ for all $1 \le j \le i\}$;
    if ($c > count[q']$) then
      begin
        $count[q'] := c$;
        $\mathcal{I}_{\mathrm{seen}} := \mathcal{I}_{\mathrm{seen}} \cup \mathcal{I}_{\mathrm{unseen\_fulfilled}}$;
        for all $i \in \mathcal{I}_{\mathrm{unseen\_fulfilled}}$ do push $(i, depth)$ on $condition\_stack$;
        $depth := depth + 1$;
        for all $t = \langle q', \Gamma', F', \{q''\} \rangle \in \Delta$ such that $\Gamma' \neq \emptyset$ and $q'' \in visited$ do
          SECOND-SEARCH($t$);
        $depth := depth - 1$;
        $(i, d)$ := topmost element of $condition\_stack$;
        while ($d = depth$) do
          begin
            $\mathcal{I}_{\mathrm{seen}} := \mathcal{I}_{\mathrm{seen}} \setminus \{i\}$;
            pop $(i, d)$ off $condition\_stack$;
            $(i, d)$ := topmost element of $condition\_stack$;
          end
      end
  end
```

Fig. 8.2: Nested depth-first search language emptiness checking algorithm for nondeterministic automata working in inf-acceptance mode using one or more acceptance conditions. The emptiness check is started by calling the TOP-LEVEL-SEARCH procedure with the initial state q_I of the automaton as a parameter

the state q such that the chain fulfills the condition. For all $q \in Q$, $count[q]$ will never decrease during the execution of the algorithm.

$\mathcal{I}_{\text{seen}}$: A set of indices of acceptance conditions "seen" in a chain of transitions (beginning with the transition from which a second search was started at line 9), the path corresponding to which ends in the state referred to by the program variable q in a nested recursive call of the SECOND-SEARCH procedure. The conditions "seen" in this chain include also the conditions that were associated with the state from which the search was started (line 8). In every nested call to the SECOND-SEARCH procedure, the algorithm first collects the indices of all previously "unseen" acceptance conditions that are included in the acceptance conditions of the transition given as a parameter for the procedure (line 16). These indices are then added to the $\mathcal{I}_{\text{seen}}$ set later at line 21 before any nested recursive calls to the procedure.

depth: An integer variable representing the number of transitions in a chain (beginning with the transition from which a second search was started) that corresponds to a path that ends in the state referred to by the program variable q' in a nested recursive call of the SECOND-SEARCH procedure.

condition_stack: A stack used for recording a (partial) history of changes made to the $\mathcal{I}_{\text{seen}}$ set during a second search in the automaton. To simplify the presentation of the SECOND-SEARCH procedure, the stack is initialized with a sentinel element that will never be removed from the stack (and thus the stack can never be empty at lines 27 or 32).

To simplify the presentation of the algorithm, the global variables *depth* and $\mathcal{I}_{\text{seen}}$ could be modeled as parameters of the SECOND-SEARCH procedure (in which case also the variable *condition_stack* could be eliminated). However, we treat these variables as globals to facilitate the analysis of the algorithm's memory requirements.

The second search proceeds from a state q to its successor q' only if the conditions "seen" in a chain of transitions corresponding to a path that ends in the state q' include the next condition (or possibly several consecutive conditions in the acceptance condition ordering), the index of which is not yet recorded in the value of $count[q']$. At line 17, the algorithm finds the maximal index of an acceptance condition which has been "seen" along with all of its predecessors (in the acceptance condition ordering) during the second search; whether to proceed to a successor of q is then decided at line 18.

Example 8.3.1 We illustrate the behavior of the emptiness checking algorithm with a simple example. Consider the nondeterministic automaton shown in Fig. 8.3 (a) (for simplicity, we assume that all transitions in the automaton have nonempty guards and omit them from the figure because the behavior of the language emptiness checking algorithm does not depend on the contents of such guards). This automaton has eight states and three acceptance conditions f_1, f_2 and f_3 represented in the figure by $\bullet$, $\circ$ and $\bullet$, respectively. It is easy to see that the automaton consists of a single maximal strongly connected component, and thus the language inf-recognized

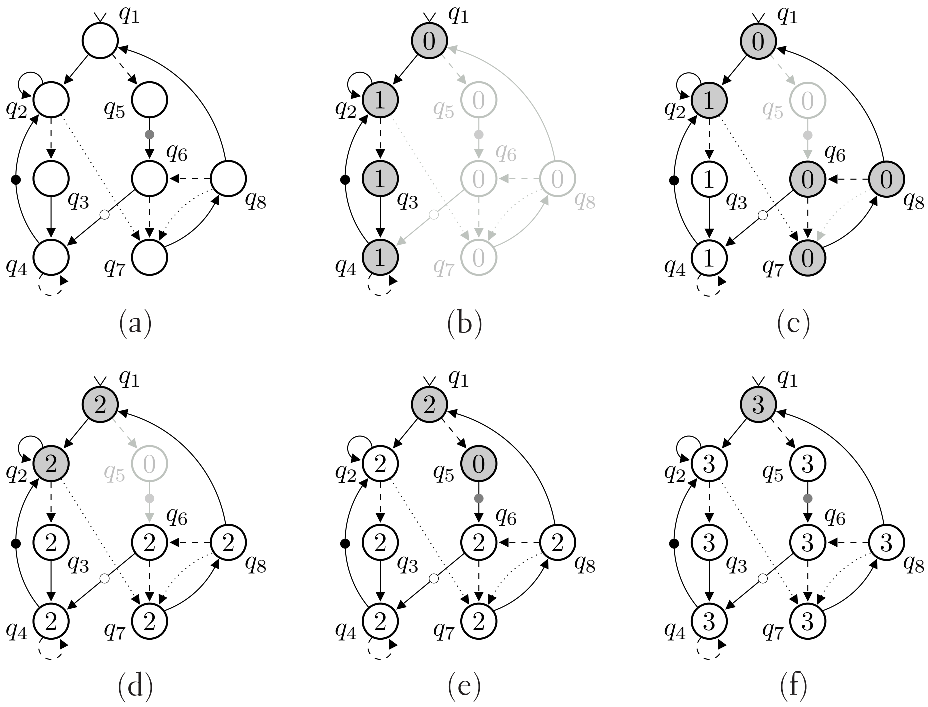


Fig. 8.3: Example on running the emptiness checking algorithm on an automaton. (a) An automaton with three acceptance conditions and a nonempty language; (b)–(f) State of the algorithm when the top-level search is about to backtrack from the states q_4, q_6, q_2, q_5 and q_1, respectively

by the automaton is not empty. We briefly describe the steps the algorithm takes to verify this property. (In the following, we assume that the algorithm examines the transitions starting from a state of the automaton in the order determined by the indices of their target states. Thus, for example, in state q_8, the algorithm first examines the transition with target state q_1, followed by the transitions with target states q_6 and q_7, respectively.)

Starting from the state q_1, the top-level search first explores the states q_2, q_3 and q_4. Before the search backtracks from the state q_4, a second search is started from each transition with source state q_4. Because the transition with target state q_2 is an f_1- (•-)transition, the second search starting from this transition explores the states q_2, q_3 and q_4 and updates $count[q_2]$, $count[q_3]$ and $count[q_4]$ to 1. The search will not proceed from q_2 to q_7, however, because the top-level search did not visit the state q_7 before this second search. Figure 8.3 (b) represents the state of the algorithm when the top-level search is about to backtrack from the state q_4 (the contents of the *count* table is represented by numbers in the states; the shaded states represent the contents of the top-level search stack).

The top-level search now backtracks from q_4 and q_3 (without making further changes to the *count* table in second searches), and then explores the states q_7, q_8 and q_6. Even though the transition from q_6 to the state q_4 is an f_2-(○-)transition, the second search started from this transition will not proceed to the state q_4, because $count[q_6] < 1$ holds at the beginning of this search (and when the top-level search is about to backtrack from the state q_6, see Fig. 8.3 (c)). The contents of this table remain unchanged until a second search is started from the transition with source state q_2 and target state q_7.

Because $count[q_2] = 1 > 0 = count[q_7]$ holds at the beginning of the second search from the transition between these two states, the second search will update $count[q_7]$, $count[q_8]$, $count[q_1]$ and $count[q_6]$ to 1. Furthermore, because the transition from q_6 to q_4 is an f_2- ($\circ$-)transition, the search will revisit all descendants of q_6 explored in the top-level search, updating the *count* table as shown in Fig. 8.3 (d), which represents the state of the algorithm when the top-level search backtracks from the state q_2.

After backtracking from the state q_2, the top-level search enters the last unvisited state q_5 of the automaton. The algorithm backtracks from this state without making any changes to the *count* table in the second search started from the transition between q_5 and q_6; see Fig. 8.3 (e).

A second search is started once more from the transition from q_1 to q_5. The state of the algorithm after this second search is shown in Fig. 8.3 (f). Because q_1 is still on the top-level search stack and $count[q_1] = 3$ holds at the end of this search, the algorithm reports that the language inf-recognized by the automaton is not empty. ■

8.3.1 Resource Requirements

Clearly, the *count* table associates each state of the automaton with an integer that will never exceed the number of acceptance conditions $|\mathcal{F}|$ in the automaton. Because entering a state q' during a second search (line 19) implies incrementing the value of $count[q']$, it is easy to see that the algorithm needs at most $|\mathcal{F}|+1$ passes of the state space of the automaton. It is also easy to see that the top-level search stack will contain at most $|Q|$ elements at any time during the search. The implicit recursion stack used for nested calls of the SECOND-SEARCH procedure will grow to contain at most $|Q| \cdot |\mathcal{F}| + 1$ elements in the worst case. If the *count* table is represented as a hash table indexed with state descriptors, each of which consumes s bits of memory, the table has to store $|Q| \cdot \big(s + \lceil \log_2(|\mathcal{F}| + 1) \rceil\big)$ bits in the worst case. (The *visited* set can be combined with this table by inserting states to the table as they are entered in the top-level search.)

The $\mathcal{I}_{\text{seen}}$ set can be represented as a bit vector with $|\mathcal{F}|$ bits. Because elements are added to *condition_stack* only when there are conditions whose indices are still missing from the $\mathcal{I}_{\text{seen}}$ set, it is easy to see that this stack will never contain more than $|\mathcal{F}| + 1$ elements. Note that this bound would not be as obvious if the *depth* and $\mathcal{I}_{\text{seen}}$ variables had been modeled as parameters of the SECOND-SEARCH procedure: in this case the information stored in *condition_stack* would be implicit in the activation records of nested recursive procedure calls (and would amount to $|Q| \cdot |\mathcal{F}| + 1$ instead of $|\mathcal{F}| + 1$ elements in the worst case). It is straightforward to check that the integer *depth*, the $\mathcal{I}_{\text{seen}}$ set and the elements in the *condition_stack* stack need $O\big(|\mathcal{F}| \log_2(|Q| \cdot |\mathcal{F}|)\big)$ bits for their representation in the worst case.

Additionally, the search procedures need (in a standard way) to keep track of information on how to generate the next transition starting from a state in the loops at lines 5–11 and 24–25 such that the information is preserved over nested recursive calls of the procedures.

Table 8.1 lists worst-case resource requirements (number of bits consumed by a hash table used to implement a search and the number of states visited

Table 8.1: Upper bounds for complexity measures for checking the language inf-recognized by a nondeterministic with n states and $1 \leq m < \omega$ acceptance conditions for emptiness; s_{g} and s_{d} are the numbers of bits required for representing state descriptors in the original and degeneralized automaton, respectively ($s_{\mathrm{g}} \leq s_{\mathrm{d}}$).

Emptiness checking strategy	Number of entries in hash table	Memory required for hash table (bits)	Number of states visited
explicit degeneralization with classic NDFS [Godefroid and Holzmann 1993]	nm	$nm(s_{\mathrm{d}} + 2)$	$2nm$
classic NDFS with on-the-fly degeneralization [Bošnački 2003]	n	$n(s_{\mathrm{g}} + 2m)$	$2nm$
generalized nested search [Tauriainen 2004, 2006]	n	$n(s_{\mathrm{g}} + m)$	$n(m+1)$
generalized NDFS (Fig. 8.2; [Tauriainen 2005])	n	$n\big(s_{\mathrm{g}} + \lceil \log_2(m+1) \rceil\big)$	$n(m+1)$

in the search in the worst case) for several language emptiness checking algorithms based on the nested depth-first search algorithm of Courcoubetis et al. [1991, 1992] when applied to nondeterministic automata working in inf-acceptance mode using multiple acceptance conditions. (For simple experimental comparison of generalized search against nongeneralized algorithms, see [Tauriainen 2006].)

The original nested depth-first search algorithm of Courcoubetis et al. [1991, 1992] for nondeterministic automata with a single acceptance condition needs two passes of the state space of the automaton in the worst case to decide whether the automaton inf-recognizes the empty language (for a description of the basic nested depth-first search algorithm, see, for example, the textbook [Clarke et al. 1999]). Godefroid and Holzmann [1993] suggested an implementation of the classic algorithm that stores two bits of information per each state of the automaton in a hash table indexed with state descriptors of the automaton to keep track of the two passes in which a state may have been visited. The worst-case number of bits needed for the hash table and number of states visited by this algorithm follow directly from the result that degeneralizing an automaton with n states and $1 \leq m < \omega$ acceptance conditions may result in an m-fold increase in the number of states of the automaton in the worst case (Proposition 8.2.1). As shown by Bošnački [2003], combining degeneralization with the search algorithm (called "on-the-fly degeneralization" in Table 8.1) makes it possible to implement the search using less memory by observing that the states in the degeneralized automaton are copies of states of the original automaton (in which case the two additional bits associated with each of the m different copies of a state can be stored into the same entry of the hash table).

The worst-case memory requirements can be improved further by bypassing degeneralization, in which case also the worst-case number of states visited during the search can be reduced [Tauriainen 2004, 2006]; ordering the acceptance conditions allows for a further reduction in the number of bits that need to be stored in the search hash table [Tauriainen 2005]. The algorithm in Fig. 8.2 corresponds to this variant of the generalized nested search algorithm from [Tauriainen 2004, 2006] enhanced with an optimization suggested by Couvreur et al. [2005] to speed up the detection of language nonemptiness by initiating second searches immediately after processing a transition at line 7 (instead of waiting until all transitions starting from a state have been processed as done in the classic NDFS algorithm).

8.3.2 Correctness of the Algorithm

In this section we prove the correctness of the algorithm.

Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ ($\mathcal{F} \neq \emptyset$) be a nondeterministic automaton (working in inf-acceptance mode) given as input for the algorithm. It is easy to see that the algorithm will start a second search at line 9 from every transition (with a nonempty guard), the source state of which is either the initial state q_I of the automaton, or a state reachable from q_I via a chain of transitions with nonempty guards, and the second search is started exactly once from each of these transitions. Therefore the transitions are processed in some well-defined order: we write $t \leq t'$ ($t, t' \in \Delta$) to indicate that the second search from t is started at line 9 before the second search from t' (or $t = t'$). (Other inequalities between transitions are defined in the obvious way.) Additionally, we write $visited(t)$ and $path\_stack(t)$ to refer to the contents of the *visited* set and the top-level search stack (respectively) at the beginning of a second search from the transition t; note that these data structures remain unchanged in all recursive calls to the SECOND-SEARCH procedure. We also write $path\_stack(q)$ to denote the contents of the top-level search stack at line 13 of the algorithm when the top-level search is about to backtrack from a state $q \in Q$.

It is easy to check (by induction on the number of nested recursive calls of the SECOND-SEARCH procedure) that, if $depth = d$ and $\mathcal{I}_{\text{seen}} = \mathcal{I}$ hold for an integer d and a set of indices of acceptance conditions $\mathcal{I} \subseteq \{1, 2, \ldots, |\mathcal{F}|\}$ when the algorithm is about to enter the loop at line 24, then $depth$ and $\mathcal{I}_{\text{seen}}$ will have these same values at the beginning of each iteration of the loop.

Soundness

In this section we prove the soundness of the algorithm (with respect to checking for the nonemptiness of a language inf-accepted by an automaton). We first formalize a simple fact about states in the top-level search stack.

Lemma 8.3.2 *Let q and q' be states in $path\_stack$ such that $q = q'$ holds, or q was pushed before q' on the stack. There exists a path from q to q' in $\mathcal{A}$ such that the path corresponds to a (possibly empty) chain of transitions with nonempty guards in $\mathcal{A}$.*

Proof: The result holds trivially if $q = q'$. Otherwise q' was pushed on the stack in a recursive call of the top-level search procedure (started at line 7 when q was on top of the stack), and it follows by induction that there exists a path from q to q' in the automaton such that this path corresponds to a chain of transitions with nonempty guards. $\square$

The soundness of the algorithm is based on the observation that every state in the top-level search stack known to be reachable via a chain of transitions (with nonempty guards) that fulfills an acceptance condition is actually in a cycle that fulfills the condition.

Lemma 8.3.3 *Let $t \in \Delta$ be a transition from which the algorithm starts a second search at line 9. If $count[q] \geq n$ holds for some $q \in path\_stack(t)$ and $1 \leq n \leq |\mathcal{F}|$ during a second search started at line 9 from the transition t, then the automaton contains a cycle of transitions with nonempty guards which fulfills the acceptance condition f_n and visits the state q.*

<u>Proof:</u> We claim that there exists an integer $1 \leq k < \omega$, states $q_0, q_1, \ldots, q_k \in Q$ (with $q_0 = q$) and transitions $t_0, t_1, \ldots, t_k \in \Delta$ such that for all $0 \leq i < k$, q_i and q_{i+1} are reachable from each other in the automaton via nonempty chains of transitions with nonempty guards, $t_i \geq t_{i+1}$ holds for all $0 \leq i < k$, and q_{k-1} is reachable from q_k via a chain of transitions (with nonempty guards) which fulfills the acceptance condition f_n. Clearly, if such sequences exist, then q_0 $(= q)$ is a source state of a transition in a cycle of transitions with nonempty guards which fulfills the acceptance condition f_n, and the result follows.

We prove the claim by constructing sequences with the required properties. Let $q_0 \stackrel{def}{=} q$, and let $t_0 \stackrel{def}{=} t$. Because $count[q_0] \geq n > 0$ holds during the second search from t_0, there exists a transition $t_1 \in \Delta$, $t_1 \leq t_0$, such that $count[q_0]$ was updated for the first time to a value greater than or equal to n during a second search started at line 9 from the transition t_1. Let $q_1 \in Q$ be the source state of t_1.

Assume that q_i and t_i have been defined for some $1 \leq i < \omega$ such that $count[q_{i-1}]$ is updated for the first time to a value greater than or equal to n during a second search started at line 9 from the transition $t_i \leq t_{i-1}$. If $count[q_i] \geq n > 0$ already holds at the beginning of this second search, there exists a transition $t' \in \Delta$, $t' < t_i$, such that $count[q_i]$ was updated for the first time to a value greater than or equal to n during a second search started from the transition t' (with source state $q' \in Q$). In this case we let $q_{i+1} \stackrel{def}{=} q'$ and $t_{i+1} \stackrel{def}{=} t'$ and continue the inductive construction.

By repeating the construction, we obtain a sequence of states $q_0, q_1, q_2, \ldots$ and a sequence of transitions $t_0 \geq t_1 > t_2 > \cdots$ such that for all $i = 0, 1, 2, \ldots$, $count[q_i]$ was updated for the first time to a value greater than or equal to n in a second search started from the transition t_{i+1} with source state q_{i+1}. Because the automaton is finite, the sequence of transitions is finite, and because $count[q'] = 0$ initially holds for all $q' \in Q$, there exists an index k such that $count[q_k] < n$ holds at the beginning of a second search from the transition t_k.

Let $0 \leq i < k$. Because $count[q_i]$ is updated for the first time to a value greater than or equal to n during a second search started from the transition $t_{i+1} \leq t_i$, q_i is reachable from the source state q_{i+1} of t_{i+1} via a nonempty chain of transitions with nonempty guards, and $q_i \in visited(t_{i+1})$ holds. On the other hand, because $t_{i+1} \leq t_i$ holds, the top-level search could not have backtracked from the state q_i before backtracking from q_{i+1}, and thus $q_i \in path\_stack(t_{i+1})$ holds. Because q_{i+1} is on top of $path\_stack$ during the second search from t_{i+1}, it follows by Lemma 8.3.2 that the automaton contains a path from q_i to q_{i+1} that corresponds to a (possibly empty) chain of transitions with nonempty guards. It follows that the states q_i and q_{i+1} are reachable from each other in the automaton via nonempty chains of transitions whose guards are nonempty.

Because $count[q_k] < n$ holds at the beginning of the second search from t_k, $n \notin \mathcal{I}_{\text{seen}}$ holds at line 16 when the second search procedure is called at line 9 with t_k as a parameter. Because $count[q_{k-1}]$ is nevertheless updated to a value greater than or equal to n during the second search from t_k (at line 20), it is easy to see that n must be inserted to $\mathcal{I}_{\text{unseen\_fulfilled}}$ during the search, and thus the second search from t_k must reach q_{k-1} via a chain of transitions (with nonempty guards) which fulfills the acceptance condition f_n. Therefore q_{k-1} and q_k are source states of transitions in a cycle that fulfills the condition f_n, and the result follows. $\square$

It is now easy to prove the soundness of the algorithm using Lemma 8.3.3.

Proposition 8.3.4 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ ($\mathcal{F} \neq \emptyset$) be the nondeterministic automaton (working in inf-acceptance mode) given as input for the algorithm. Let $t \in \Delta$ be a transition from which the algorithm starts a second search at line 9. If $count[q] = |\mathcal{F}|$ holds during this search for a state $q \in path\_stack(t)$, then $\mathcal{A}$ contains an accepting cycle. In particular, this holds if the algorithm reports that $\mathcal{A}$ does not inf-recognize the empty language.*

Proof: Because the top-level search is started from the initial state q_I of the automaton, $q_I \in path\_stack(t)$ certainly holds, and there exists a path from q_I to q in $\mathcal{A}$ (corresponding to a chain of transitions, the guards of which are nonempty) by Lemma 8.3.2. Because $count[q] \geq i$ holds for all $1 \leq i \leq |\mathcal{F}|$, it follows by Lemma 8.3.3 that for all $1 \leq i \leq |\mathcal{F}|$, the automaton contains a cycle (of transitions with nonempty guards) which fulfills the acceptance condition f_i and visits the state q. Because all of these cycles visit the state q, they can be merged together to obtain an accepting cycle for the automaton. The soundness of the algorithm now follows from the condition at line 10 of the top-level search procedure since the program variable q refers to the topmost state of $path\_stack$ at this point. $\square$

Completeness

We now turn to the completeness of the algorithm. Again, we start by listing several basic facts about the behavior of the algorithm for future reference.

Lemma 8.3.5 *Let $t = \langle q, \Gamma, F, \{q'\} \rangle \in \Delta$ be a transition from which the algorithm starts a second search at line 9. Then, $q' \in visited(t)$ holds.*

Proof: The result follows from the condition at line 7 of the algorithm: if $q' \notin visited$ holds at this point, the algorithm calls the TOP-LEVEL-SEARCH procedure recursively for the state q', and q' is added to the $visited$ set at line 4 of the algorithm before a second search from the transition t. $\square$

Lemma 8.3.6 *Let $t \in \Delta$ be a transition from which the algorithm starts a second search at line 9. If there exists a state $q \in visited(t)$ and a transition $t' = \langle q, \Gamma, F, \{q'\} \rangle \in \Delta$ such that $\Gamma \neq \emptyset$ and $q' \in Q \setminus visited(t)$ hold, then $q \in path\_stack(t)$ holds.*

Proof: Let q be a state satisfying the assumptions. Because $q \in visited(t)$ holds, the top-level search has entered q. If the search had also backtracked from q, then the algorithm would have reached line 12 with q on top of the top-level search stack. Therefore the algorithm would have initiated a second search from all transitions (with a nonempty guard) having q as its source state, in particular, from the transition t'. But then $q' \in visited$ would hold at the beginning of the search from t by Lemma 8.3.5, contrary to the assumption. Therefore, the top-level search has not yet backtracked from q, and $q \in path\_stack(t)$ holds. □

Lemma 8.3.7 *Let $\langle q, \Gamma, F, \{q'\} \rangle \in \Delta$ be a transition for which the algorithm calls the second search procedure at line 9 or 25. If $c \geq n$ holds at line 18 for some $0 \leq n \leq |\mathcal{F}|$ at the beginning of the call, then $count[q'] \geq n$ holds when this call returns.*

Proof: If $count[q'] < n$ holds at line 18, $count[q']$ is updated to the value c $(\geq n)$ at line 20 of the algorithm, and the result follows from the fact that $count[q']$ never decreases during the execution of the algorithm. □

Lemma 8.3.8 *Assume that the algorithm reaches line 20 during a second search started from a transition $t \in \Delta$ such that $c \geq n$ holds for some $1 \leq n \leq |\mathcal{F}|$, and the program variable q' refers to a state $q \in Q$ at this point. The second search procedure will be called recursively for every transition (with a nonempty guard) having q as its source state such that $c \geq n$ holds at line 18 at the beginning of each call.*

Proof: The result holds trivially if q has no successors, or if all transitions starting from q have an empty guard. Otherwise, let $t' = \langle q, \Gamma, F, \{q'\} \rangle$ be a transition (with $\Gamma \neq \emptyset$) having q as its source state. It is easy to see from line 20 that $count[q] \geq n$ will hold at the end of the second search from t.

Assume that $q' \in visited(t)$ holds. Denote by $\mathcal{I}$ the contents of the $\mathcal{I}_{\text{seen}}$ set at the beginning of the first iteration of the loop at line 24 when the program variable q' refers to the state q with $c \geq n$. Because $c \geq n$ holds, it is easy to see that $\{1, 2, \ldots, n\} \subseteq \mathcal{I}$ holds at this point. Because $\mathcal{I}_{\text{seen}} = \mathcal{I}$ holds at the beginning of each iteration of the loop (and therefore, in particular, at the beginning of the recursive call of the second search procedure for the transition t'), line 17 guarantees that $c \geq n$ holds at line 18 in the beginning of this call.

If $q' \notin visited(t)$ holds, the algorithm has not yet started a second search from the transition t' (Lemma 8.3.5), and thus $t' > t$ holds. Furthermore, $q \in path\_stack(t)$ holds by Lemma 8.3.6, i.e., the top-level search has not backtracked from the state q before the second search from t. Clearly, a second search is started from the transition t' before the top-level search backtracks from q. Because $count[q] \geq n$ holds at the end of the second search from t, it follows from the initialization of the $\mathcal{I}_{\text{seen}}$ set at line 8 that $c \geq n$

will hold at line 18 when the second search is started from the transition t' at line 9. $\square$

In the following results, we shall refer to a maximal strongly connected component $C \subseteq Q \cup \Delta$ that contains either the initial state of the automaton or a state reachable from the initial state of the automaton via a chain of transitions with nonempty guards. Clearly, the top-level search will in this case explore all states in the component, and thus there exists a state $\hat{q} \in C \cap Q$ that is the first state of C pushed on the top-level search stack. Because the elements of this stack are accessed in "last in, first out" order, it is easy to see that $\hat{q}$ is the last state of C from which the top-level search backtracks.

Our goal is to show that if the component C contains an accepting cycle, then there exists a state $q \in C \cap Q$ in the component (namely, the first state $\hat{q}$ of the component entered in the top-level search) for which $count[q] = |\mathcal{F}|$ holds when the top-level search is about to backtrack from the state q. Clearly, this implies that the algorithm will report that the language inf-recognized by the automaton is not empty (at the latest) at line 10 with $\hat{q}$ on top of $path\_stack$. (Because C contains an accepting cycle, C contains at least one transition having a nonempty guard and $\hat{q}$ as its source state, and thus it is easy to see that the algorithm will in fact reach line 10 with $\hat{q}$ on top of the top-level search stack before the top-level search backtracks from $\hat{q}$.)

More generally, we shall be interested in finding states $q \in C \cap Q$ for which $count[q] \geq n$ holds for some $1 \leq n \leq |\mathcal{F}|$ when the top-level search is about to backtrack from the state q at line 13. Given a path in the automaton (corresponding to a nonempty chain of transitions with nonempty guards), this property is guaranteed to hold for the last state q (in the path) for which $count[q] \geq n$ holds at the end of a second search started in the first state of the path unless the path ends in the state q.

Lemma 8.3.9 *Let $q_0, q_1, \ldots, q_k \in Q$ be the list of consecutive states in a path (corresponding to a nonempty chain of transitions with nonempty guards) from the state q_0 to the state q_k in the automaton for some $1 \leq k < \omega$. Assume that there exists a maximal index $0 \leq \ell \leq k$ for which $count[q_\ell] \geq n$ holds for some $1 \leq n \leq |\mathcal{F}|$ at the end of a second search from a transition $t \in \Delta$ having q_0 as its source state (line 10). If $\ell < k$ holds, then q_ℓ is a state for which $count[q_\ell] \geq n$ already holds when the top-level search backtracks from the state q_ℓ. Actually, in this case $count[q_\ell] \geq n$ holds before the algorithm starts a second search from any transition from q_ℓ to $q_{\ell+1}$.*

Proof: Because $n \geq 1$ holds, there exists a transition $t' \in \Delta$, $t' \leq t$, such that the second search started from t' reached line 20 of the algorithm such that the program variable q' referred for the first time to the state q_ℓ with $c \geq n$. Clearly, $q_\ell \in visited(t')$ holds, and the top-level search had entered q_ℓ at some point. By Lemma 8.3.8, the algorithm calls the second search procedure recursively for every transition (with a nonempty guard) with source state q_ℓ and target state $q_{\ell+1}$ such that $c \geq n$ holds at line 18 at the beginning of this call. If $q_{\ell+1} \in visited(t')$ holds, such a call occurs during the second search started from the transition t'. But then $count[q_{\ell+1}] \geq n$ would hold at the end of the call by Lemma 8.3.7, and therefore also at the end of the second search from t. This contradicts the maximality of ℓ, however. It follows that $q_{\ell+1} \notin \mathit{visited}(t')$ holds, and by Lemma 8.3.6, $q_\ell \in \mathit{path\_stack}(t')$ holds. Thus the top-level search backtracks from the state q_ℓ after the second search from t', and $\mathit{count}[q_\ell] \geq n$ holds when this occurs. The second claim follows from Lemma 8.3.5: because $q_{\ell+1} \notin \mathit{visited}(t')$ holds, no second search can have been started at line 9 from a transition with source state q_ℓ and target state $q_{\ell+1}$ before updating $\mathit{count}[q_\ell]$ to a value greater than or equal to n (in the second search from t'). $\square$

Lemma 8.3.9 leads to the following result, which can then be used to show that the reachability information (i.e., the knowledge on the reachability of states via chains of transitions that fulfill certain acceptance conditions) stored in the *count* table propagates towards the first state of the maximal strongly connected component C entered in the top-level search.

Lemma 8.3.10 *Let C be a maximal strongly connected component of the automaton, and let $\hat{q} \in C \cap Q$ be the first state of the component entered in the top-level search. Let $q \in C \cap Q$, $q \neq \hat{q}$, be another state in the component such that $\mathit{count}[q] \geq n$ holds for some $1 \leq n \leq |\mathcal{F}|$ when the top-level search is about to backtrack from the state q at line 13. There exists a state $q' \in C \cap Q$, $q' \neq q$, such that the top-level search backtracks from q' after backtracking from q, and $\mathit{count}[q'] \geq n$ holds when this occurs at line 13.*

<u>Proof:</u> Because q and $\hat{q}$ belong to the same maximal strongly connected component, but $q \neq \hat{q}$ holds, there exists a path (corresponding to a nonempty chain of transitions with nonempty guards) from q to $\hat{q}$ in the automaton. Clearly, all states in this path are contained in C. Let $q_0, q_1, \ldots, q_k \in C \cap Q$ ($1 \leq k < \omega$, $q_0 = q$, $q_k = \hat{q}$) be the list of consecutive states in this path. Because $\mathit{count}[q_0] \geq n$ holds when the top-level search is about to backtrack from the state q_0 at line 13, there exists a maximal index $0 \leq \ell \leq k$ such that $\mathit{count}[q_\ell] \geq n$ holds at this point. Clearly, $\mathit{count}[q_\ell] \geq n$ holds already at the end of a second search started at line 9 from the last transition $t \in \Delta$ (with source state q_0 and a nonempty guard) which was processed in the loop between lines 5 and 11 (and such a transition exists because C contains a path from q_0 to q_k corresponding to a nonempty chain of transitions with nonempty guards).

If $\ell = k$ holds, the result follows immediately (with $q' = \hat{q}$) by the choice of $\hat{q}$. Otherwise, by Lemma 8.3.9, $\mathit{count}[q_\ell] \geq n$ holds at the beginning of any second search starting at line 9 from a transition from q_ℓ to $q_{\ell+1}$ (and still when the top-level search is about to backtrack from the state q_ℓ). Let $t' \in C \cap \Delta$ be a transition from q_ℓ to $q_{\ell+1}$ with a nonempty guard. If the top-level search had backtracked from q_ℓ before backtracking from q_0 (or if $q_\ell = q_0$), a second search would have been started from the transition t'. But then, because $\mathit{count}[q_\ell] \geq n$ holds at the beginning of this second search (and because $q_{\ell+1} \in \mathit{visited}(t')$ necessarily holds by Lemma 8.3.5 at this point), it follows by Lemma 8.3.7 that $\mathit{count}[q_{\ell+1}] \geq n$ would hold when the top-level search backtracks from the state q_0. This, however, contradicts the maximality of ℓ. Therefore, $q_\ell \neq q_0$ holds, and the top-level search backtracks from q_ℓ only after backtracking from q_0 $(= q)$. The result now follows with $q' = q_\ell$. $\square$

Corollary 8.3.11 *Let C be a maximal strongly connected component of the automaton, and let $\hat{q} \in C \cap Q$ be the first state of the component entered in the top-level search. Let $q \in C \cap Q$ be a state in the component such that $count[q] \geq n$ holds for some $1 \leq n \leq |\mathcal{F}|$ when the top-level search is about to backtrack from q at line 13. Then, $count[\hat{q}] \geq n$ holds when the top-level search is about to backtrack from $\hat{q}$.*

Proof: The result holds trivially if $q = \hat{q}$. Let thus $q \neq \hat{q}$. Because $count[q] \geq n \geq 1$ holds when the top-level search is about to backtrack from the state q at line 13, then, by Lemma 8.3.10, there exists a state $q' \in C \cap Q$, $q' \neq q$, from which the top-level search backtracks after backtracking from the state q, and $count[q'] \geq n$ holds when this occurs.

By repeating the argument, we can now construct a sequence of distinct states $q, q', \ldots \in C \cap Q$ such that for any state q'' in the sequence, the top-level search backtracks from the state q'' only after backtracking from all states that precede it in the sequence, and $count[q''] \geq n$ holds when the top-level search backtracks from q''. Because C is finite and $\hat{q}$ is the last state in C from which the top-level search backtracks, the sequence ends with the state $\hat{q}$, and the result follows. $\square$

On the other hand, when the top-level search is about to backtrack from the first state $\hat{q}$ of the component C entered in the top-level search, the following relationship holds between $count[\hat{q}]$ and the values stored in the $count$ table for other states in the component.

Lemma 8.3.12 *Let C be a maximal strongly connected component of the automaton, and let $\hat{q} \in C \cap Q$ be the first state of the component entered in the top-level search. Assume that $count[\hat{q}] = n$ holds for some $0 \leq n \leq |\mathcal{F}|$ when the top-level search backtracks from $\hat{q}$ at line 13. Then, $count[q] \geq n$ holds for all $q \in C \cap Q$ at this point.*

Proof: The result holds trivially if $n = 0$. For the rest of the proof, we assume that $n > 0$ holds. Let $q \in C \cap Q$. Because q and $\hat{q}$ belong to the same maximal strongly connected component, there exists a path from $\hat{q}$ to q in the automaton such that the path corresponds to a (possibly empty) chain of transitions with nonempty guards. We proceed by induction on the number of transitions in the chain.

If the chain is empty, then $q = \hat{q}$, and the result holds for the state q trivially.

Assume that the result holds for all states in C to which there exists a path from $\hat{q}$ corresponding to a shortest chain of $0 \leq k < \omega$ transitions with nonempty guards, and let $q \in C \cap Q$ be a state that is reachable from $\hat{q}$ via a shortest chain $(t_i)_{i=0}^{k}$ of $k + 1$ transitions with nonempty guards ($t_i = \langle q_i, \Gamma_i, F_i, \{q_{i+1}\} \rangle \in \Delta$ for all $0 \leq i \leq k$, $q_0 = \hat{q}$, and $q_{k+1} = q$). Clearly, q_k is reachable from $\hat{q}$ via a path that corresponds to a chain of k transitions with nonempty guards, and because $\hat{q}, q \in C$, $q_k \in C$ holds also. By the induction hypothesis, $count[q_k] \geq n > 0$ holds when the top-level search backtracks from $\hat{q}$. This implies the existence of a transition $t \in \Delta$ such that $count[q_k]$ was first updated to a value greater than or equal to n during a second search started (at line 9) from the transition t. It follows that the algorithm reached line 20 such that the program variable q' referred to the state q_k with $c \geq n$. Let $t' \in C \cap \Delta$ be a transition (with a nonempty guard) from q_k to q. By Lemma 8.3.8, the algorithm will call the second search procedure recursively for the transition t' such that $c \geq n$ holds at line 18 at the beginning of this call. Furthermore, by the choice of $\hat{q}$, this call occurs before the top-level search backtracks from the state $\hat{q}$. By Lemma 8.3.7, it follows that $count[q] \geq n$ holds when the top-level search backtracks from $\hat{q}$.

The result now follows by induction for all states $q \in C \cap Q$. □

A maximal strongly connected component C that contains an accepting cycle includes an f-transition (with a nonempty guard) for every acceptance condition $f \in \mathcal{F}$. We wish to use Corollary 8.3.11 to show that the existence of such transitions forces the condition at line 10 to hold in the first state of C entered in the top-level search. However, Corollary 8.3.11 does not directly refer to such f-transitions. We therefore need the following technical result, which establishes a connection between the existence of an f_n-transition (in C) for some $1 \leq n \leq |\mathcal{F}|$ and a state $q \in C \cap Q$ for which $count[q] \geq n$ holds when the top-level search is about to backtrack from the state.

Lemma 8.3.13 *Let C be a maximal strongly connected component of the automaton, and let $\hat{q} \in C \cap Q$ be the first state of the component entered in the top-level search. Let $t_{f_{n+1}} = \langle q_{f_{n+1}}, \Gamma_{f_{n+1}}, F_{f_{n+1}}, \{q'_{f_{n+1}}\} \rangle \in C \cap \Delta$ be an f_{n+1}-transition of $\mathcal{A}$ (with $f_{n+1} \in F_{f_{n+1}}$ and $\Gamma_{f_{n+1}} \neq \emptyset$) for some $0 \leq n < |\mathcal{F}|$. Assume that, before the top-level search backtracks from the state $\hat{q}$, the* SECOND-SEARCH *procedure is called at line 9 or 25 of the algorithm with $t_{f_{n+1}}$ as a parameter such that $c \geq n$ holds at the beginning of this call at line 18. There exists a state $q \in C \cap Q$ such that $count[q] \geq n + 1$ holds when the top-level search is about to backtrack from the state q at line 13.*

Proof: Clearly, the call to the SECOND-SEARCH procedure with $t_{f_{n+1}}$ as a parameter occurs during a second search started at line 9 from a transition $t \in \Delta$ such that there exists a path (corresponding to a possibly empty chain of transitions with nonempty guards) from the source state of t to the state $q_{f_{n+1}}$ in the automaton. Because $c \geq n$ holds at line 18 and $f_{n+1} \in F_{f_{n+1}}$, it actually follows that $c \geq n+1$ holds at line 18, and thus $count[q'_{f_{n+1}}] \geq n+1$ holds at the end of the second search from t by Lemma 8.3.7.

Because $q'_{f_{n+1}} \in C$ holds, there exists a path from $q'_{f_{n+1}}$ to $\hat{q}$ that corresponds to a (possibly empty) chain of transitions with nonempty guards that visits only states in C. Thus there exists a path from the source state q_0 of t to $\hat{q}$ (corresponding to a nonempty chain of transitions whose guards are nonempty) in the automaton. Let $q_0, q_1, \ldots, q_k \in Q$ ($1 \leq k < \omega$, $q_k = \hat{q}$, $q_i = q'_{f_{n+1}}$ for some $1 \leq i \leq k$, and $q_j \in C$ for all $i \leq j \leq k$) be the list of consecutive states in this path. Because $count[q'_{f_{n+1}}] \geq n + 1$ holds at the end of the second search from t, there exists a maximal index $i \leq \ell \leq k$ such that $count[q_\ell] \geq n + 1$ holds at this point.

If $\ell = k$ holds, then the result follows (with $q = \hat{q}$) by the choice of $\hat{q}$. Otherwise $count[q_\ell] \geq n + 1$ holds when the algorithm is about to backtrack from the state q_ℓ by Lemma 8.3.9, and the result holds with $q = q_\ell$ because $q_\ell \in C$. □

We can now prove the completeness of the algorithm.

Proposition 8.3.14 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ (where $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is a nonempty finite set of $1 \leq n < \omega$ acceptance conditions) be a nondeterministic automaton (working in inf-acceptance mode) given as input for the algorithm. If $\mathcal{A}$ contains an accepting cycle, the algorithm reports that the language inf-recognized by $\mathcal{A}$ is not empty.*

Proof: Because the automaton $\mathcal{A}$ contains an accepting cycle, $\mathcal{A}$ has a maximal strongly connected component $C \subseteq Q \cup \Delta$ that includes a state reachable from q_I via a chain of transitions with nonempty guards, and an f_i-transition (with a nonempty guard) for all $1 \leq i \leq |\mathcal{F}|$. Let $\hat{q}$ be the first state of C entered in the top-level search.

Assume that $count[\hat{q}] = m < n$ holds when the top-level search is about to backtrack from the state $\hat{q}$. By Lemma 8.3.12, it follows that $count[q] \geq m$ holds for all $q \in C \cap Q$ at this point.

Because C contains an accepting cycle, there exists an f_{m+1}-transition $t_{m+1} \in C \cap \Delta$ with a nonempty guard. We claim that the algorithm calls the second search procedure at line 9 or 25 for the transition t_{m+1} such that $c \geq m$ holds at line 18 at the beginning of this call. This is clear if $m = 0$, because a second search is started from every transition in C before the top-level search backtracks from the state $\hat{q}$. If $m > 0$, then, because $count[q] \geq m$ holds for all $q \in C \cap Q$ before the top-level search backtracks from the state $\hat{q}$ (but $count[q] = 0$ holds for all $q \in C \cap Q$ initially), the algorithm reached line 20 such that the program variable q' referred to the source state of t_{m+1} with $c \geq m$ before the top-level search backtracks from $\hat{q}$. In this case the second search procedure will be called for the transition t_{m+1} (before the top-level search backtracks from $\hat{q}$) such that $c \geq m$ holds at line 18 at the beginning of this call by Lemma 8.3.8.

Because $c \geq m$ holds at line 18 when the second search procedure is called with the transition t_{m+1} (which is an f_{m+1}-transition of $\mathcal{A}$) as a parameter, it now follows by Lemma 8.3.13 that there exists a state $q_{m+1} \in C \cap Q$ such that $count[q_{m+1}] \geq m + 1$ holds before the top-level search backtracks from q_{m+1}. But then, by Corollary 8.3.11, $count[\hat{q}] \geq m + 1$ holds when the top-level search backtracks from the state $\hat{q}$. This contradicts the assumption that $count[\hat{q}] = m < n$ holds at this point.

It follows that $count[\hat{q}] \geq n$ necessarily holds when the top-level search backtracks from $\hat{q}$ (and $count[\hat{q}] = n$, because $c \leq n$ clearly holds always at line 20, the only location where the value of $count[\hat{q}]$ may change). Therefore, because $\hat{q} \in C$ is the source state of at least one transition with a nonempty guard, the algorithm will report that the language inf-recognized by the automaton is not empty at line 10 before the top-level search backtracks from $\hat{q}$ (at the latest, after a second search from the last transition examined in the loop between lines 5 and 11 with $\hat{q}$ on top of $path\_stack$). □

8.3.3 Compatibility with Enhancements of Classic Nested Depth-First Search

This section lists some observations [Tauriainen 2005, 2006] on the compatibility of the language emptiness checking algorithm with techniques used with the classic nested depth-first search algorithm. Many of these techniques are designed to be used in the specific context of verification, where the nondeterministic automata are built on-the-fly as "products" of several smaller components. The exact form of the product operation depends on the semantics of the components, which may not be strictly defined as automata. A classic example of a product operation is the *synchronous product* of a finite number ($1 \leq n < \omega$) of nondeterministic automata $\mathcal{A}_i = \langle \Sigma_i, Q_i, \Delta_i, q_{Ii}, \mathcal{F}_i \rangle$ ($1 \leq i \leq n$) defined as the automaton $\mathcal{A} \stackrel{\text{def}}{=} \big\langle \bigcup_{1\leq i \leq n} \Sigma_i, Q_1 \times Q_2 \times \cdots \times Q_n, \Delta, \langle q_{I1}, q_{I2}, \ldots, q_{In} \rangle, \bigcup_{1 \leq i \leq n} \big(\mathcal{F}_i \times \{i\}\big) \big\rangle$, where

$$\Delta \stackrel{\text{def}}{=} \Big\{ \big\langle \langle q_1, q_2, \ldots, q_n \rangle, \bigcap_{1 \leq i \leq n} \Gamma_i, \bigcup_{1 \leq i \leq n} \big(F_i \times \{i\}\big), \{\langle q_1', q_2', \ldots, q_n' \rangle\} \big\rangle \;\Big|\; \big\langle q_i, \Gamma_i, F_i, \{q_i'\} \big\rangle \in \Delta_i \text{ for all } 1 \leq i \leq n \Big\}.$$

If the automata $\mathcal{A}_i$ share the same alphabet Σ, it is straightforward to check that the automaton $\mathcal{A}$ inf-accepts a word $w \in \Sigma^\omega$ iff $w \in \bigcap_{1 \leq i \leq n} \mathcal{L}_{\text{inf}}(\mathcal{A}_i)$ holds. (As illustrated by the definition of the synchronous product, the number of transitions generated by product constructions is often exponential in the number of components in the product. This feature provides motivation for minimizing the number of states explored by a language emptiness checking algorithm [Schwoon and Esparza 2005].)

Constructing Accepting Cycles

In addition to checking the languages inf-recognized by nondeterministic automata for emptiness, verification usually includes the additional task of finding accepting cycles for automata with nonempty languages to be used as "counterexamples" for showing why a verification run failed. Unlike the classic nested depth-first search algorithm [Courcoubetis et al. 1991, 1992], the emptiness checking algorithm of Fig. 8.2 does not support extracting accepting cycles directly from its internal data structures when reporting language nonemptiness (except for a path from the initial state of the automaton to a state visited by an accepting cycle). In principle, it is straightforward to construct an actual accepting cycle using additional search in the automaton, for example, by using techniques suggested by Latvala and Heljanko [1999, 2000]. (This search can be restricted to states explored in the emptiness search algorithm.)

Optimizations

A standard heuristic to speed up the detection of language nonemptiness is to use information about the states in the top-level search stack to abort a second search as soon as possible [Holzmann et al. 1997]: it is easy to see from Lemma 8.3.3 that language nonemptiness can be reported immediately when a second search updates $count[q]$ to the value $|\mathcal{F}|$ at line 20 of the algorithm for a state q currently in the top-level search stack. States in the stack can be distinguished easily by using an additional bit of memory per state in the search hash table [Holzmann et al. 1997].

It is easy to show that the second search started from a transition could always be restricted to the maximal strongly connected component (or any overapproximation thereof) which contains the source state of the transition: if the source state and the target state of a transition t belong to different maximal strongly connected components, it follows that no state in the top-level search stack can be reached via a chain of transitions beginning with the transition t (and thus $count[q]$ will not change for any state q in the top-level search stack when exploring states reachable via the transition t). Although it does not make sense to find the maximal strongly connected components of the automaton in order to run the NDFS algorithm, partial information on the components may nevertheless be available if the automaton is built as the product of smaller component automata. As suggested by Edelkamp et al. [2001, 2004], the often more easily computable information on the maximal strongly connected components of the component automata can be used to limit the extent of second searches in the NDFS algorithm.

The algorithm in Fig. 8.2 enhanced with the above optimization from [Holzmann et al. 1997] is compatible also with a heuristic suggested by Couvreur et al. [2005] to further speed up the detection of language nonemptiness. Intuitively, whereas the enhanced algorithm considers only the acceptance conditions "seen" in a chain of transitions beginning from the source state of a second search when testing whether a state reached in the top-level search stack closes an accepting cycle, it is sometimes possible to detect language nonemptiness faster by considering also the conditions in a chain of transitions from this state to the state currently on the top of the top-level search stack [Couvreur et al. 2005].

Partial Order Reduction

The effort needed for state space exploration in verification (language emptiness checking) algorithms can often be reduced significantly by identifying parts of the state space which can be ignored in the search without compromising the completeness of the algorithm. For example, using concepts such as independence between transitions (defined in terms of the semantics of the components from which the automaton is built), the classic nested depth-first search algorithm can be combined with on-the-fly partial order reduction [Peled 1994, 1996] to restrict the set of successors to explore from a given state of the state space. From a purely theoretical viewpoint, partial order reduction can be seen as an automaton transformation which preserves the emptiness or nonemptiness of its language; in principle, a language emptiness checking algorithm can be combined with any such transformation to obtain a sound and complete verification procedure.

In practice, however, combining state space reductions with language emptiness checking may require additional effort to ascertain that the reduction will not interfere with the preservation of language (non)emptiness. For example, in the case of partial order reduction and classic nested depth-first search, extra memory may be needed to ensure that revisiting a state during a second search will not affect the set of successors chosen for the state when it was first visited [Holzmann et al. 1997]. The techniques suggested in [Holzmann et al. 1997] apply also to the algorithm in Fig. 8.2 to make it compatible with partial order reduction.

Bitstate Hashing and Hash Compaction

Explicit-state model checking tools employ probabilistic state exploration techniques such as *bitstate hashing* [Holzmann 1987, 1988] or *hash compaction* [Wolper and Leroy 1993; Stern and Dill 1995, 1996] to reduce memory usage at the risk of losing the completeness of the language emptiness checking algorithm by exploring only a part of the state space of the automaton, using hash functions that may map several states of the automaton to the same entry in the search hash table. Compatibility with bitstate hashing was originally considered to be one of the main advantages of the classic nested depth-first search algorithm [Courcoubetis et al. 1991, 1992] over alternative language emptiness checking algorithms, although recent research on these algorithms has shown bitstate hashing to be applicable as well to algorithms based on Tarjan's algorithm [Geldenhuys and Valmari 2005]. Below, we list changes needed to make the language emptiness checking algorithm presented in this chapter compatible with imperfect hashing.

The soundness of the language emptiness checking algorithm shown in Fig. 8.2 rests on the property that $count[q] \geq n$ holds for some state q currently in the top-level search stack for some $1 \leq n \leq |\mathcal{F}|$ only if the automaton contains a cycle which visits the state q and fulfills the acceptance condition f_n (Lemma 8.3.3). Unfortunately, this property cannot be guaranteed with imperfect hashing without a way to keep exact track of the contents of the $count$ table for states currently in the top-level search stack. A simple possibility is to store the values associated with the states in the stack in a separate hash table that is always consulted first when accessing entries in the $count$ table during second searches. When a state q is entered in the top-level search, $count[q]$ should be set to zero in this hash table; conversely, updates made to this table should be copied back to the imperfectly hashed part of the table when the top-level search backtracks from the state q. (To ensure that the algorithm still works within the bound given in Table 8.1 for the number of states visited by the algorithm, the update should not be done, however, if the value of $count[q]$ would decrease in the imperfectly hashed part of the table.)

The feasibility of the above strategy in the reduction of memory requirements depends critically on the assumption that the depth of the top-level search stack remains small in comparison with the number of states in the automaton. Unfortunately, there exists experimental evidence [Pelánek 2004] suggesting that this assumption does not usually hold for depth-first state exploration algorithms in actual verification examples.

State Space Caching

Restricting the second searches to states that have been explored in the top-level search makes the generalized algorithm incompatible with *state space caching* techniques [Holzmann 1985; Godefroid et al. 1993, 1995], which reduce the memory requirements of the search at the risk of repeating the search multiple times in parts of the automaton; the completeness of the algorithm is not preserved, for example, in the extreme case when only the states in the top-level search stack are kept in the state space cache. This also makes it impossible to apply heuristics for choosing which states to keep in the set of visited states (see, e.g., [Brim et al. 2001]). Making the algorithm compatible with state space caching therefore limits opportunities for removing states from the *visited* set. However, the requirements on the *count* table can be relaxed somewhat: for example, entries that do not refer to states currently in the top- or second-level search stacks need not be kept in this table. (Making this work nevertheless requires ensuring that, when backtracking to a state during a second search, the iteration over the transitions beginning from the state at line 24 is never continued from a transition that has already been processed.)

9 CONCLUSION

The close connection between linear time temporal logic and self-loop alternating automata gives rise to conceptually simple translation procedures between LTL and automata. To implement decision procedures for this logic and its applications, the alternating automata built from formulas in the logic are usually translated explicitly or implicitly into nondeterministic automata, which can then be analyzed using, for example, well-known simple graph algorithms. The worst-case exponential combinatorial cost of nondeterminization makes the behavior of the decision procedures very sensitive to the properties of the alternating automata: intuitively, even small changes to the alternating automata may have visible effects on the results of nondeterminization. This motivates the study of methods for optimizing the translation of LTL into self-loop alternating automata. Even though the nondeterministic automata built from self-loop alternating automata still have exponential size in the length of an LTL specification in the worst case, the translation from LTL into nondeterministic automata may be manageable in a more robust way as separate, more easily understandable subtasks, as evidenced also by practical observations on the difficulty of implementing efficient translation procedures for LTL correctly [Tauriainen and Heljanko 2000, 2002]. Additionally, it is in some cases possible to speed up on-the-fly verification by combining nondeterminization of alternating automata directly with language emptiness checking [Hammer et al. 2005].

Although the differences between state-based and transition-based, and on the other hand, nongeneralized and generalized, Büchi acceptance are simple in a theoretical sense, this work illustrates and reinforces in a single framework the well-known advantages of using generalized transition-based acceptance for both understanding and even implementing the LTL verification procedure [Couvreur 1999; Gastin and Oddoux 2001; Giannakopoulou and Lerda 2002; Thirioux 2002]. The constructions involving self-loop alternating automata built from LTL formulas benefit from the generalized notion of acceptance: the acceptance conditions of the automata can easily be used for encoding semantic properties of LTL formulas ([Gerth et al. 1995; Gastin and Oddoux 2001]; Sect. 3.1.1), and transition-based acceptance allows separating purely "structural" features of the constructions from the semantics of acceptance. For example, generalized transition-based acceptance allows the basic state-transition structure of nondeterministic automata built from self-loop alternating automata to be defined, in effect, by adapting the classic subset construction ([Gastin and Oddoux 2001]; Theorem 4.2.1). Although this opportunity is not unique to using the transition-based notion of acceptance [Hammer et al. 2005], the state-based construction of Hammer et al. [2005] nevertheless applies only to a subclass of very weak alternating automata, whereas our transition-based approach covers all self-loop alternating automata, however, at the—in the general case, unavoidable—cost of increasing the number of acceptance conditions in the subset construction (Proposition 4.2.3). However, the introduction of new acceptance conditions can be avoided when working with automata that have uniform acceptance synchronized accepting runs on all words in their language (Theorem 4.3.2).

Most importantly, all automata built from LTL formulas using the translation rules presented in this work (Sect. 3.1, Table 3.1, Table 5.2, Table 5.3) have this property (by Proposition 4.3.5, via Lemma 4.3.8, Proposition 5.4.3, and Proposition 5.5.1); in comparison, the construction of Hammer et al. [2005] depends on using additional rewrite rules for LTL formulas to ensure that the automata belong to the intended subclass.

Also syntactic optimization techniques proposed for direct translation algorithms from LTL into nondeterministic automata [Daniele et al. 1999; Giannakopoulou and Lerda 2002] can easily be described as modifications to the nondeterminization construction (Sect. 4.5). Further improvements to the translation procedure are easily understood by making use of information on language containment relationships between self-loop alternating automata (Sect. 5.3); these relationships were essential also in the design of the refined translation rules that can be used as heuristics to reduce universal choice in the automata (Table 5.2, Table 5.3). In most cases, the optimized nondeterminization construction remains applicable also when removing redundant transitions from the automata (Sect. 6.2.5). Because LTL and self-loop alternating automata have the same expressive power ([Rohde 1997; Löding and Thomas 2000]; Theorem 3.3.2, Theorem 3.4.2), the language containment based improvements to the translation procedure can, in principle, be implemented effectively without a general complementation procedure for alternating automata—in many cases, by simply reusing the basic translation procedure from LTL into self-loop alternating automata.

Transition-based acceptance is useful also for extending known syntactic subclasses of LTL [Schneider 1999; Maidl 2000a] (shown to be expressively closely related in Sect. 4.6.4 of this work) which can be translated directly into self-loop nondeterministic automata without applying the subset construction (Sect. 4.6, Sect. 5.6.3). For example, even though there is no very weak nondeterministic automaton (in the sense of Rohde [1997] or Gastin and Oddoux [2001], i.e., using state-based acceptance) recognizing the models of the LTL formula $\mathsf{GF}p$ for an atomic proposition p, transition-based acceptance allows formulas of this form (and, as a matter of fact, also their logical conjunctions) to be naturally expressed as single-state self-loop automata, which can furthermore be obtained effectively using the refined translation rules (Table 5.2, Table 5.3). Additionally, the automata-theoretic study of these subclasses of LTL leads easily to simple **NP**-completeness results on their satisfiability (Corollary 4.6.11, Proposition 5.6.7). These syntactic subclasses differ from classic subclasses whose satisfiability is **NP**-complete by restricting the ways to combine subformulas into compound formulas instead of the set of operators allowed to occur in the formulas.

Finally, generalized transition-based acceptance can also be handled directly in language emptiness checking algorithms for nondeterministic automata without the need to translate the automata to simpler formalisms [Couvreur 1999; Tauriainen 2004, 2006]. In particular, because the blow-up caused by translating generalized acceptance into nongeneralized acceptance cannot be avoided in the general case (Proposition 8.2.1), bypassing degeneralization of automata leads to a new variant of the nested depth-first search language emptiness checking algorithm of Courcoubetis et al. [1991, 1992] (Sect. 8.3) with improved worst-case resource requirements under generalized acceptance (see Table 8.1). The author has contributed an implementation of the search heuristic used in this algorithm to version 0.3 of the open source model checking library Spot [Duret-Lutz and Poitrenaud 2004].

Directions for Further Work

Implementing and extending the translation procedure. The first obvious task for further work is to implement the high-level translation procedure presented in Ch. 7 to evaluate its feasibility in practice. On a more theoretical level, the translation procedure should be extended with translation rules for past time LTL connectives. There are freely available tools for testing the correctness of implementations of translation procedures from future time LTL into automata [Tauriainen and Heljanko 2000, 2002]; such tools can also be used for benchmarking purposes.

"Top-down" vs. "bottom-up" translation. Unlike tableau-based translation algorithms, the proposed construction for translating LTL into alternating automata does not support on-demand definition of an automaton starting from its initial state [Gerth et al. 1995; Bollig and Leucker 2001, 2003]; obviously, a "top-down" version of the translation procedure would be very attractive for being directly combined with on-the-fly nondeterminization or emptiness checking techniques. This feature is not unique to the proposed translation, however, as the need for constructing an automaton in full is also implicit, for example, in the design of simulation-based minimization techniques for alternating and nondeterministic automata. Furthermore, because the number of states in an alternating automaton built from an LTL formula is linear in the length of the formula, the translation from LTL into alternating automata is intuitively less prone to exhibit the worst-case exponential space requirements than direct explicit translation algorithms from LTL into nondeterministic automata. A task for further work is to see whether the translation procedure could be implemented in a top-down manner while retaining the applicability of at least some of the techniques used to minimize an automaton during the translation.

Explicit vs. symbolic definitions for alternating automata. Many of the proposed structural optimizations to self-loop alternating automata depend on the explicit notion of a transition of an alternating automaton. As a matter of fact, the explicit encoding for the transition relation of an automaton is rather unattractive in a complexity-theoretic sense: for example, because complementing an alternating automaton may result in an exponential blow-up in the automaton's explicit representation, polynomial space automata constructions which depend on the possibility to complement automata in polynomial space do not trivially remain so when using the explicit definition. Similarly, the basic translation procedure from LTL into self-loop alternating automata has exponential worst-case space complexity in the length of LTL specifications. Of course, this feature is common to translation procedures for constructing automata from LTL formulas explicitly. An obvious task for further work is to investigate whether transition-based generalized acceptance and any of the explicit techniques for removing transitions could be combined with (or explained in terms of) traditional Boolean encodings of the transition relation.

Cost of improvements to the translation procedure. Most of the proposed improvements to the translation procedure make use of language containment tests between self-loop alternating automata. Even though the language emptiness checks in the high-level algorithm of Ch. 7 can be performed in exponential space in essentially a single application of a nondeterminization construction, the practicability of this procedure is nevertheless likely to depend critically on heuristics to ensure that the language containment testing is applied only conservatively due to the high worst-case resource requirements. As a matter of fact, it is a valid question whether LTL formulas used in verification applications (which often have only few temporal operators with shallow nesting of temporal subformulas) exhibit redundancies which display as language containment between subformulas. This question can be answered only with experimental evaluation of the translation procedure. Some of the suggested techniques are nevertheless applicable without the need for testing for language containment and could therefore be used for improving related translation procedures that do not rely on such tests. For example, combining the universally applicable refined translation rules with standard techniques for removing redundant transitions from automata sometimes reduces the number of transitions in the generated automata from exponential to polynomial in the length of a formula for certain families of LTL formulas (Ex. 6.2.12, Ex. 6.2.13).

Applications of acceptance synchronized runs. Despite the simple form of the results obtained by arguing about uniform acceptance synchronized runs of alternating automata (for example, the optimized universal subset construction of Theorem 4.3.2 and the refined translation rule for the $\wedge$ connective in Table 5.2), checking for the preservation of the conditions that imply the existence of such runs (Proposition 4.3.5) in optimizations for alternating automata is in the general case rather tedious due to the strong semantic nature of the conditions. (This is apparent already in the proofs of Lemma 5.4.8, Lemma 5.5.4 and Proposition 6.2.8.) For the same reason, even though the result on the applicability of the optimized universal subset construction (Corollary 4.3.7) does not explicitly refer to self-loop alternating automata (only acceptance closure and the existence of representative states for acceptance), it is an open question whether there actually exist any other easily characterizable subclasses of alternating automata which could benefit from the optimized nondeterminization construction. Identifying such subclasses would directly give new insight into the problem posed by Kupferman and Vardi [2004] of searching for subclasses of alternating automata which support optimized translation into nondeterministic automata with generalized acceptance.

Improved heuristics for detecting redundant transitions. The techniques suggested for removing redundant transitions from alternating automata (Proposition 6.2.2, Proposition 6.2.3) are obviously limited due to their applicability only to the initial transitions of self-loop alternating automata. Of course, this choice of focus stems from the goal of reducing the effort needed to apply translation rules in the translation procedure from LTL into self-loop alternating automata. A possible task for further research is to investigate whether the results could be generalized or extended to obtain a complete "local" characterization of redundant (initial) transitions of self-loop or more general alternating automata. Such generalizations would likely have to be based on a weakened notion of substitutability of transitions under fin-acceptance: similarly to distributing the individual elements in the guard of a substituted transition among a set of substituting transitions, also the set intersection of the languages recognized by the subautomata rooted at the target states of the substituted transition could be divided among several substituting transitions. (In principle, an initial transition of any alternating automaton is redundant iff Proposition 2.3.15 still holds after removing the transition for all words for which it held before removing the transition.) On the other hand, designing efficient heuristics for searching for suitable sets of transitions in Proposition 6.2.2 and Proposition 6.2.3 is already a nonobvious task with the definition of substitutability used in this work; a weaker definition of substitutability will likely complicate the search for suitable subsets of transitions. Another possibility for further work is to evaluate the feasibility of transition redundancy analysis in self-loop alternating automata by making use of the connection to linear time temporal logic via the reverse translation procedure.

Reducing nondeterminism in automata built in translation. Some authors [Etessami and Holzmann 2000; Sebastiani and Tonetta 2003] have questioned whether minimizing the size of a nondeterministic automaton built from a logical specification actually reduces the resources needed for verification in practice: clearly, minimizing the nondeterministic automaton only affects the worst case. As a matter of fact, there is some research and experimental evidence [Thirioux 2002; Latvala 2003; Sebastiani and Tonetta 2003] to support the view that the combinatorial explosion that arises in the actual verification step may be more efficiently reduced in practice by replacing nondeterministic choice with deterministic choice in the automata. There are some obvious possibilities for trying to increase determinism in the suggested constructions: for example, the translation rules for the binary temporal operators could be based on stronger fixpoint characterizations [Couvreur 1999; Thirioux 2002]. Additionally, a new translation rule for the $\vee$ connective (modeled after the refined $\wedge$ rule) could possibly help in "postponing" nondeterministic choice in branches of runs of automata.

Extending NP-complete syntactic subclasses of LTL. Obviously, the syntactic subclasses of LTL (LTL^{CND} and $LTL^{\mathrm{CND}^{+}}$) which were shown to be translatable into nondeterministic self-loop automata without applying the subset construction leave room for further possible extensions: the subclasses were defined by examining only the translation rules without considering any of the other suggested optimizations, such as removing transitions from automata. On the other hand, as seen already from the definition of the class $LTL^{\mathrm{CND}^{+}}$ (Sect. 5.6.3), purely syntactic characterizations of such subclasses may easily degenerate into complex lists of mutually recursive special cases. Alternatively, the translation could be used in an experimental approach to determining the fraction of LTL formulas (from a finite set of LTL formulas restricted, for example, by their node size) belonging to such subclasses. A comparison of the subclass (or any of the other known **NP**-complete subclasses of LTL) against formula patterns used in verification [Dwyer et al. 1999] could also give new insight into the theoretical complexity of verification of different types of properties.

BIBLIOGRAPHY

S. Aggarwal, C. Courcoubetis, and P. Wolper. Adding liveness properties to coupled finite-state machines. *ACM Transactions on Programming Languages and Systems*, 12(2):303–339, 1990.

M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. In *Proceedings of the 8th Annual ACM Symposium on Principles of Programming Languages (POPL 1981)*, pages 164–176. Association for Computing Machinery, 1981.

M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. *Acta Informatica*, 20(3):207–226, 1983.

S. Ben-David, R. Bloem, D. Fisman, A. Griesmayer, I. Pill, and S. Ruah. Automata construction algorithms optimized for PSL. Deliverable 3.2/4, PROSYD, 2005.

O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Proceedings of the 6th International Conference on Computer Aided Verification (CAV 1994)*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155. Springer-Verlag, 1994.

B. Bollig and M. Leucker. Deciding LTL over Mazurkiewicz traces. In *Proceedings of the 8th International Symposium on Temporal Representation and Reasoning (TIME 2001)*, pages 189–197. IEEE Computer Society, 2001.

B. Bollig and M. Leucker. Deciding LTL over Mazurkiewicz traces. *Data & Knowledge Engineering*, 44(2):219–238, 2003.

D. Bošnački. A nested depth first search algorithm for model checking with symmetry reduction. In *Proceedings of the 22nd IFIP WG6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2002)*, volume 2529 of *Lecture Notes in Computer Science*, pages 65–80. Springer-Verlag, 2002.

D. Bošnački. A light-weight algorithm for model checking with symmetry reduction and weak fairness. In *Proceedings of the 10th International* SPIN *Workshop on Model Checking Software (SPIN 2003)*, volume 2648 of *Lecture Notes in Computer Science*, pages 89–103. Springer-Verlag, 2003.

L. Brim, I. Černá, and M. Nečesal. Randomization helps in LTL model checking. In *Proceedings of the Joint Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM-PROBMIV 2001)*, volume 2165 of *Lecture Notes in Computer Science*, pages 105–119. Springer-Verlag, 2001.

R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

J. A. Brzozowski and E. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theoretical Computer Science*, 10 (1):19–35, 1980.

J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

D. Bustan, D. Fisman, and J. Havlicek. Automata construction for PSL. Technical Report MSC05-04, The Weizmann Institute of Science, 2005.

D. Bustan and O. Grumberg. Applicability of fair simulation. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002)*, volume 2280 of *Lecture Notes in Computer Science*, pages 401–414. Springer-Verlag, 2002.

D. Bustan and O. Grumberg. Applicability of fair simulation. *Information and Computation*, 194(1):1–18, 2004.

A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

A. K. Chandra and L. J. Stockmeyer. Alternation. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS 1976)*, pages 98–108. IEEE Computer Society, 1976.

Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.

E. M. Clarke and I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 428–437. Springer-Verlag, 1989.

E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of the Workshop on Logics of Programs 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1982a.

E. M. Clarke and E. A. Emerson. Using branching time temporal logic to syntesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982b.

E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. In *Proceedings of the 10th Annual ACM Symposium on Principles of Programming Languages (POPL 1983)*, pages 117–126. Association for Computing Machinery, 1983.

E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Proceedings of the 6th International Conference on Computer Aided Verification (CAV 1994)*, volume 818 of *Lecture Notes in Computer Science*, pages 415–427. Springer-Verlag, 1994.

E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. *Formal Methods in System Design*, 10(1):47–71, 1997.

E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158. Association for Computing Machinery, 1971.

C. Courcoubetis, M. Y. Vardi, P. Wolper, and M. Yannakakis. Memory-efficient algorithms for the verification of temporal properties. In *Proceedings of the 2nd International Conference on Computer-Aided Verification (CAV 1990)*, volume 531 of *Lecture Notes in Computer Science*, pages 233–242. Springer-Verlag, 1991.

C. Courcoubetis, M. Y. Vardi, P. Wolper, and M. Yannakakis. Memory-efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275–288, 1992.

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In *Proceedings of the FM'99 World Congress on Formal Methods in the Development of Computing Systems, Volume I*, volume 1708 of *Lecture Notes in Computer Science*, pages 253–271. Springer-Verlag, 1999.

J.-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. On-the-fly emptiness checks for generalized Büchi automata. In *Proceedings of the 12th International* SPIN *Workshop on Model Checking Software (SPIN 2005)*, volume 3639 of *Lecture Notes in Computer Science*, pages 169–184. Springer-Verlag, 2005.

M. Daniele, F. Giunchiglia, and M. Y. Vardi. Improved automata generation for linear temporal logic. In *Proceedings of the 11th International Conference on Computer Aided Verification (CAV 1999)*, volume 1633 of *Lecture Notes in Computer Science*, pages 249–260. Springer-Verlag, 1999.

G. G. de Jong. An automata theoretic approach to temporal logic. In *Proceedings of the 3rd International Conference on Computer Aided Verification (CAV 1991)*, volume 575 of *Lecture Notes in Computer Science*, pages 477–487. Springer-Verlag, 1992.

S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1998)*, volume 1373 of *Lecture Notes in Computer Science*, pages 61–72. Springer-Verlag, 1998.

S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.

A. Duret-Lutz and D. Poitrenaud. SPOT: an extensible model checking library using transition-based generalized Büchi automata. In *Proceedings of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2004)*, pages 76–83. IEEE Computer Society, 2004.

M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 1999 International Conference on Software Engineering (ICSE 1999)*, pages 411–420. Association for Computing Machinery, 1999.

S. Edelkamp, A. Lluch Lafuente, and S. Leue. Directed explicit model checking with HSF-SPIN. In *Proceedings of the 8th International* SPIN *Workshop on Model Checking Software (SPIN 2001)*, volume 2057 of *Lecture Notes in Computer Science*, pages 57–79. Springer-Verlag, 2001.

S. Edelkamp, S. Leue, and A. Lluch-Lafuente. Directed explicit-state model checking in the validation of communication protocols. *International Journal on Software Tools for Technology Transfer (STTT)*, 5(2–3):247–267, 2004.

E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, pages 995–1072. Elsevier, 1990.

E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time. In *Proceedings of the 10th Annual ACM Symposium on Principles of Programming Languages (POPL 1983)*, pages 127–140. Association for Computing Machinery, 1983.

E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33 (1):151–178, 1986.

E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS 1991)*, pages 368–377. IEEE Computer Society, 1991.

E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for fragments of μ-calculus. In *Proceedings of the 5th International Conference on Computer Aided Verification (CAV 1993)*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer-Verlag, 1993.

E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science*, 258(1–2): 491–522, 2001.

E. A. Emerson and A. P. Sistla. Deciding branching time logic. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC 1984)*, pages 14–24. Association for Computing Machinery, 1984a.

E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984b.

K. Etessami. Stutter-invariant languages, ω-automata, and temporal logic. In *Proceedings of the 11th International Conference on Computer Aided Verification (CAV 1999)*, volume 1633 of *Lecture Notes in Computer Science*, pages 236–248. Springer-Verlag, 1999.

K. Etessami. A hierarchy of polynomial-time computable simulations for automata. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR 2002)*, volume 2421 of *Lecture Notes in Computer Science*, pages 131–144. Springer-Verlag, 2002.

K. Etessami and G. J. Holzmann. Optimizing Büchi automata. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR 2000)*, volume 1877 of *Lecture Notes in Computer Science*, pages 153–167. Springer-Verlag, 2000.

K. Etessami, Th. Wilke, and R. A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP 2001)*, volume 2076 of *Lecture Notes in Computer Science*, pages 694–707. Springer-Verlag, 2001.

K. Etessami, Th. Wilke, and R. A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. *SIAM Journal on Computing*, 34(5):1159–1175, 2005.

C. Fritz. Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In *Proceedings of the 8th International Conference on Implementation and Application of Automata (CIAA 2003)*, volume 2759 of *Lecture Notes in Computer Science*, pages 35–48. Springer-Verlag, 2003.

C. Fritz. Concepts of automata construction from LTL. In *Proceedings of the 12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2005)*, volume 3835 of *Lecture Notes in Computer Science*, pages 728–742, 2005.

C. Fritz and Th. Wilke. State space reductions for alternating Büchi automata: Quotienting by simulation equivalences. In *Proceedings of the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2002)*, volume 2556 of *Lecture Notes in Computer Science*, pages 157–168. Springer-Verlag, 2002.

C. Fritz and Th. Wilke. Simulation relations for alternating Büchi automata. *Theoretical Computer Science*, 338(1–3):275–314, 2005.

D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th Annual ACM Symposium on Principles of Programming Languages (POPL 1980)*, pages 163–173. Association for Computing Machinery, 1980.

P. Gastin, R. Meyer, and A. Petit. A (non-elementary) modular decision procedure for LTrL. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS 1998)*, volume 1450 of *Lecture Notes in Computer Science*, pages 356–365. Springer-Verlag, 1998.

P. Gastin, P. Moro, and M. Zeitoun. Minimization of counterexamples in SPIN. In *Proceedings of the 11th International* SPIN *Workshop on Model Checking Software (SPIN 2004)*, volume 2989 of *Lecture Notes in Computer Science*, pages 92–108. Springer-Verlag, 2004.

P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer-Verlag, 2001.

P. Gastin and D. Oddoux. LTL with past and two-way very-weak alternating automata. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS 2003)*, volume 2747 of *Lecture Notes in Computer Science*, pages 439–448. Springer-Verlag, 2003.

M. C. W. Geilen. On the construction of monitors for temporal logic properties. *Electronic Notes in Theoretical Computer Science*, 55(2), 2001.

J. Geldenhuys and A. Valmari. Tarjan's algorithm makes on-the-fly LTL verification more efficient. In *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 205–219. Springer-Verlag, 2004.

J. Geldenhuys and A. Valmari. More efficient on-the-fly LTL verification with Tarjan's algorithm. *Theoretical Computer Science*, 345(1):60–82, 2005.

R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the 15th IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification (PSTV 1995)*, pages 3–18. Chapman & Hall, 1995.

D. Giannakopoulou and F. Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Proceedings of the 22nd IFIP WG6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2002)*, volume 2529 of *Lecture Notes in Computer Science*, pages 308–326. Springer-Verlag, 2002.

P. Godefroid and G. J. Holzmann. On the verification of temporal properties. In *Proceedings of the IFIP TC6/WG6.1 13th International Symposium on Protocol Specification, Testing and Verification (PSTV 1993)*, pages 109–124. North-Holland, 1993.

P. Godefroid, G. J. Holzmann, and D. Pirottin. State-space caching revisited. In *Proceedings of the 4th International Conference on Computer Aided Verification (CAV 1992)*, volume 663 of *Lecture Notes in Computer Science*, pages 178–191. Springer-Verlag, 1993.

P. Godefroid, G. J. Holzmann, and D. Pirottin. State-space caching revisited. *Formal Methods in System Design*, 7(3):227–241, 1995.

Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC 1982)*, pages 60–65. Association for Computing Machinery, 1982.

S. Gurumurthy, F. Somenzi, and R. Bloem. Fair simulation minimization. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 610–624. Springer-Verlag, 2002.

M. Hammer, A. Knapp, and S. Merz. Truly on-the-fly LTL model checking. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2005)*, volume 3440 of *Lecture Notes in Computer Science*, pages 191–205. Springer-Verlag, 2005.

G. J. Holzmann. Tracing protocols. *AT&T Technical Journal*, 64(10):2413–2433, 1985.

G. J. Holzmann. On limits and possibilities of automated protocol analysis. In *Proceedings of the IFIP WG6.1 7th International Conference on Protocol Specification, Testing and Verification (PSTV 1987)*, pages 339–344. North-Holland, 1987.

G. J. Holzmann. An improved protocol reachability analysis technique. *Software – Practice and Experience*, 18(2):137–161, 1988.

G. J. Holzmann, D. Peled, and M. Yannakakis. On nested depth first search. In *Proceedings of the 2nd* SPIN *Workshop (SPIN 1997)*, volume 32 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

A. Isli. Mapping an LPTL formula into a Büchi alternating automaton accepting its models. In *Temporal Logic: Proceedings of the ICTL Workshop*, pages 85–90. Research Report MPI-I-94-230, Max-Planck-Institut für Informatik, 1994.

A. Isli. Converting a Büchi alternating automaton to a usual nondeterministic one. *Sādhānā – Academy Proceedings in Engineering Sciences*, 21(2): 213–228, 1996.

Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A decision algorithm for full propositional temporal logic. In *Proceedings of the 5th International Conference on Computer Aided Verification (CAV 1993)*, volume 697 of *Lecture Notes in Computer Science*, pages 97–109. Springer-Verlag, 1993.

Y. Kesten, A. Pnueli, and L. Raviv. Algorithmic verification of linear temporal logic specifications. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP 1998)*, volume 1443 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1998.

D. Kozen. On parallelism in Turing machines. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS 1976)*, pages 89–97. IEEE Computer Society, 1976.

O. Kupferman, N. Piterman, and M. Y. Vardi. Extended temporal logic revisited. In *Proceedings of the 12th International Conference on Concurrency Theory (CONCUR 2001)*, volume 2154 of *Lecture Notes in Computer Science*, pages 519–535. Springer-Verlag, 2001.

O. Kupferman and M. Y. Vardi. On the complexity of branching modular model checking. In *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR 1995)*, volume 962 of *Lecture Notes in Computer Science*, pages 408–422. Springer-Verlag, 1995.

O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. In *Proceedings of the 5th Israel Symposium on Theory of Computing and Systems (ISTCS 1997)*, pages 147–158. IEEE Computer Society, 1997.

O. Kupferman and M. Y. Vardi. An automata-theoretic approach to modular model checking. *ACM Transactions on Programming Languages and Systems*, 22(1):87–128, 2000.

O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.

O. Kupferman and M. Y. Vardi. From complementation to certification. In *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 591–606. Springer-Verlag, 2004.

O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS 2002)*, pages 383–392. IEEE Computer Society, 2002.

T. Latvala. Efficient model checking of safety properties. In *Proceedings of the 10th International* SPIN *Workshop on Model Checking Software (SPIN 2003)*, volume 2648 of *Lecture Notes in Computer Science*, pages 74–88. Springer-Verlag, 2003.

T. Latvala and K. Heljanko. Coping with strong fairness – On-the-fly emptiness checking for Streett automata. In *Proceedings of the Workshop on Concurrency, Specification and Programming (CS&P 1999)*, pages 107–118. Warsaw University, 1999.

T. Latvala and K. Heljanko. Coping with strong fairness. *Fundamenta Informaticae*, 43(1–4):175–193, 2000.

E. Leiss. Succinct representation of regular languages by Boolean automata. *Theoretical Computer Science*, 13(3):323–330, 1981.

O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages (POPL 1985)*, pages 97–107. Association for Computing Machinery, 1985.

O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.

P. Lindsay. On alternating ω-automata. *Journal of Computer and System Sciences*, 36(1):16–24, 1988.

C. Löding. *Methods for the Transformation of ω-Automata: Complexity and Connection to Second Order Logic*. Diploma thesis, Christian-Albrechts-University of Kiel, 1998.

C. Löding and W. Thomas. Alternating automata and logics over infinite words. In *Proceedings of the IFIP International Conference on Theoretical Computer Science – Exploring New Frontiers of Theoretical Informatics (IFIP TCS2000)*, volume 1872 of *Lecture Notes in Computer Science*, pages 521–535. Springer-Verlag, 2000.

M. Maidl. The common fragment of CTL and LTL. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*, pages 643–652. IEEE Computer Society, 2000a.

M. Maidl. *Using Model Checking for System Verification*. Doctoral dissertation, Fakultät für Mathematik und Informatik, Ludwig-Maximilians-Universität, München, 2000b.

Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems – Specification*. Springer-Verlag, 1992.

Z. Manna and H. B. Sipma. Alternating the temporal picture for safety. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP 2000)*, volume 1853 of *Lecture Notes in Computer Science*, pages 429–450. Springer-Verlag, 2000.

Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. In *Proceedings of the Workshop on Logics of Programs 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 253–281. Springer-Verlag, 1982.

Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 6(1):68–93, 1984.

R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966.

S. Merz and A. Sezgin. Emptiness of linear weak alternating automata. Technical report, LORIA, 2003.

M. Michel. Computation of temporal operators. *Logique et Analyse*, 28 (110–111):137–152, 1985.

S. Miyano and T. Hayashi. Alternating finite automata on ω-words. In *Proceedings of the 9th International Colloquium on Trees in Algebra and Programming (CAAP 1984)*, pages 195–209. Cambridge University Press, 1984a.

S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32(3):321–330, 1984b.

D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata, the weak monadic theory of the tree, and its complexity. In *Proceedings of the 13th International Colloquium on Automata, Languages and Programming (ICALP 1986)*, volume 226 of *Lecture Notes in Computer Science*, pages 275–283. Springer-Verlag, 1986.

D. E. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proceedings of the 3rd Annual Symposium on Logic in Computer Science (LICS 1988)*, pages 422–427. IEEE Computer Society, 1988.

D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata, the weak monadic theory of trees and its complexity. *Theoretical Computer Science*, 97(2):233–244, 1992.

D. E. Muller and P. E. Schupp. Alternating automata on infinite objects, determinacy and Rabin's theorem. In *Automata on Infinite Words*, volume 192 of *Lecture Notes in Computer Science*, pages 100–107. Springer-Verlag, 1985.

D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2–3):267–276, 1987.

D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141 (1–2):69–107, 1995.

A. Muscholl and I. Walukiewicz. An NP-complete fragment of LTL. In *Proceedings of the 8th International Conference on Developments in Language Theory (DLT 2004)*, volume 3340 of *Lecture Notes in Computer Science*, pages 334–344, 2004.

A. Muscholl and I. Walukiewicz. An NP-complete fragment of LTL. *International Journal of Foundations of Computer Science*, 16(4):743–753, 2005.

C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

R. Pelánek. Typical structural properties of state spaces. In *Proceedings of the 11th International* SPIN *Workshop on Model Checking Software (SPIN 2004)*, volume 2989 of *Lecture Notes in Computer Science*, pages 5–22. Springer-Verlag, 2004.

D. Peled. Combining partial order reductions with on-the-fly model-checking. In *Proceedings of the 6th International Conference on Computer Aided Verification (CAV 1994)*, volume 818 of *Lecture Notes in Computer Science*, pages 377–390. Springer-Verlag, 1994.

D. A. Peled. Combining partial order reductions with on-the-fly model checking. *Formal Methods in System Design*, 8(1):39–64, 1996.

D. Perrin and J.-E. Pin. *Infinite Words: Automata, Semigroups, Logic and Games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, pages 46–57. IEEE Computer Society, 1977.

A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13(1):45–60, 1981.

J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the 5th International Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 1982.

M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959. Reprinted (with corrections) in *Sequential Machines – Selected Papers*, pages 63–91. Addison-Wesley, 1964.

Y. S. Ramakrishna, L. K. Dillon, L. E. Moser, P. M. Melliar-Smith, and G. Kutty. An automata-theoretic decision procedure for future interval logic. In *Proceedings of the 12th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 1992)*, volume 652 of *Lecture Notes in Computer Science*, pages 51–67. Springer-Verlag, 1992a.

Y. S. Ramakrishna, P. M. Melliar-Smith, L. E. Moser, L. K. Dillon, and G. Kutty. Interval logics and their decision procedures, part I: An interval logic. *Theoretical Computer Science*, 166(1–2):1–47, 1996.

Y. S. Ramakrishna, L. E. Moser, L. K. Dillon, P. M. Melliar-Smith, and G. Kutty. An automata-theoretic decision procedure for propositional temporal logic with since and until. *Fundamenta Informaticae*, 17(3):271–282, 1992b.

G. S. Rohde. *Alternating Automata and the Temporal Logic of Ordinals*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.

S. Safra. On the complexity of ω-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS 1988)*, pages 319–327. IEEE Computer Society, 1988.

K. Schneider. Yet another look at LTL model checking. In *Proceedings of the 10th IFIP WG10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, volume 1703 of *Lecture Notes in Computer Science*, pages 321–325. Springer-Verlag, 1999.

K. Schneider. Improving automata generation for linear temporal logic by considering the automaton hierarchy. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001)*, volume 2250 of *Lecture Notes in Computer Science*, pages 39–54. Springer-Verlag, 2001.

S. Schwoon and J. Esparza. A note on on-the-fly verification algorithms. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2005)*, volume 3440 of *Lecture Notes in Computer Science*, pages 174–190. Springer-Verlag, 2005.

R. Sebastiani and S. Tonetta. “More deterministic” vs. “smaller” Büchi automata for efficient LTL model checking. In *Proceedings of the 12th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME 2003)*, volume 2860 of *Lecture Notes in Computer Science*, pages 126–140. Springer-Verlag, 2003.

A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC 1982)*, pages 159–168. Association for Computing Machinery, 1982.

A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000)*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263. Springer-Verlag, 2000.

U. Stern and D. L. Dill. Improved probabilistic verification by hash compaction. In *Proceedings of the IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME 1995)*, volume 987 of *Lecture Notes in Computer Science*, pages 206–224. Springer-Verlag, 1995.

U. Stern and D. L. Dill. A new scheme for memory-efficient probabilistic verification. In *Proceedings of the IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques and Protocol Specification, Testing and Verification (FORTE/PSTV 1996)*, pages 333–348. Kluwer, 1996.

R. S. Streett. Propositional dynamic logic of looping and converse. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC 1981)*, pages 375–383. Association for Computing Machinery, 1981.

R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1/2):121–141, 1982.

R. Tarjan. Depth-first search and linear graph algorithms. In *Conference Record of the 12th Annual Symposium on Switching and Automata Theory*, pages 114–121. IEEE Computer Society, 1971.

R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

H. Tauriainen. On translating linear temporal logic into alternating and nondeterministic automata. Research Report A83, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2003.

H. Tauriainen. Nested emptiness search for generalized Büchi automata. In *Proceedings of the 4th International Conference on Application of Concurrency to System Design (ACSD 2004)*, pages 165–174. IEEE Computer Society, 2004.

H. Tauriainen. A note on the worst-case memory requirements of generalized nested depth-first search. Research Report A96, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2005.

H. Tauriainen. Nested emptiness search for generalized Büchi automata. *Fundamenta Informaticae*, 70(1–2):127–154, 2006.

H. Tauriainen and K. Heljanko. Testing SPIN's LTL formula conversion into Büchi automata with randomly generated input. In *Proceedings of the 7th International* SPIN *Workshop on Model Checking Software (SPIN 2000)*, volume 1885 of *Lecture Notes in Computer Science*, pages 54–72. Springer-Verlag, 2000.

H. Tauriainen and K. Heljanko. Testing LTL formula translation into Büchi automata. *International Journal on Software Tools for Technology Transfer (STTT)*, 4(1):57–70, 2002.

X. Thirioux. Simple and efficient translation from LTL formulas to Büchi automata. *Electronic Notes in Theoretical Computer Science*, 66(2), 2002.

W. Thomas. Star-free regular sets of ω-sequences. *Information and Control*, 42(2):148–156, 1979.

W. Thomas. A combinatorial approach to the theory of ω-automata. *Information and Control*, 48(3):261–283, 1981.

W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, pages 133–191. Elsevier, 1990.

W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume III, pages 389–455. Springer-Verlag, 1997.

W. Thomas. Complementation of Büchi automata revisited. In *Jewels are Forever: Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 109–120. Springer-Verlag, 1999.

A. Valmari. The state explosion problem. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer-Verlag, 1998.

M. Y. Vardi. A temporal fixpoint calculus. In *Proceedings of the 15th Annual ACM Symposium on Principles of Programming Languages (POPL 1988)*, pages 250–259. Association for Computing Machinery, 1988.

M. Y. Vardi. Nontraditional applications of automata theory. In *Proceedings of the International Conference on Theoretical Aspects of Computer Software (TACS 1994)*, volume 789 of *Lecture Notes in Computer Science*, pages 575–597, 1994.

M. Y. Vardi. Alternating automata and program verification. In *Computer Science Today – Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science*, pages 471–485. Springer-Verlag, 1995.

M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, 1996.

M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the Symposium on Logic in Computer Science (LICS 1986)*, pages 332–344. IEEE Computer Society, 1986.

M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

P. Wolper. Temporal logic can be more expressive. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science (FOCS 1981)*, pages 340–348. IEEE Computer Society, 1981.

P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28(110–111):119–136, 1985.

P. Wolper. On the relation of programs and computations to models of temporal logic. In *Temporal Logic in Specification*, volume 398 of *Lecture Notes in Computer Science*, pages 75–123. Springer-Verlag, 1987.

P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science, Revised Lectures*, volume 2090 of *Lecture Notes in Computer Science*, pages 261–277. Springer-Verlag, 2001.

P. Wolper and D. Leroy. Reliable hashing without collision detection. In *Proceedings of the 5th International Conference on Computer Aided Verification (CAV 1993)*, volume 697 of *Lecture Notes in Computer Science*, pages 59–70. Springer-Verlag, 1993.

P. Wolper, M. Y. Vardi, and A. P. Sistla. Reasoning about infinite computation paths. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS 1983)*, pages 185–194. IEEE Computer Society, 1983.

HELSINKI UNIVERSITY OF TECHNOLOGY LABORATORY FOR THEORETICAL COMPUTER SCIENCE
RESEARCH REPORTS

HUT-TCS-A91	Mikko Särelä Measuring the Effects of Mobility on Reactive Ad Hoc Routing Protocols. May 2004.
HUT-TCS-A92	Timo Latvala, Armin Biere, Keijo Heljanko, Tommi Junttila Simple Bounded LTL Model Checking. July 2004.
HUT-TCS-A93	Tuomo Pyhälä Specification-Based Test Selection in Formal Conformance Testing. August 2004.
HUT-TCS-A94	Petteri Kaski Algorithms for Classification of Combinatorial Objects. June 2005.
HUT-TCS-A95	Timo Latvala Automata-Theoretic and Bounded Model Checking for Linear Temporal Logic. August 2005.
HUT-TCS-A96	Heikki Tauriainen A Note on the Worst-Case Memory Requirements of Generalized Nested Depth-First Search. September 2005.
HUT-TCS-A97	Toni Jussila On Bounded Model Checking of Asynchronous Systems. October 2005.
HUT-TCS-A98	Antti Autere Extensions and Applications of the A^* Algorithm. November 2005.
HUT-TCS-A99	Misa Keinänen Solving Boolean Equation Systems. November 2005.
HUT-TCS-A100	Antti E. J. Hyvärinen SATU: A System for Distributed Propositional Satisfiability Checking in Computational Grids. February 2006.
HUT-TCS-A101	Jori Dubrovin Jumbala — An Action Language for UML State Machines. March 2006.
HUT-TCS-A102	Satu Elisa Schaeffer Algorithms for Nonuniform Networks. April 2006.
HUT-TCS-A103	Janne Lundberg A Wireless Multicast Delivery Architecture for Mobile Terminals. May 2006.
HUT-TCS-A104	Heikki Tauriainen Automata and Linear Temporal Logic: Translations with Transition-Based Acceptance. September 2006.

ISBN 951-22-8343-3
ISSN 1457-7615