Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 83
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 83
Espoo 2003 HUT-TCS-A83

ON TRANSLATING LINEAR TEMPORAL LOGIC INTO ALTERNATING AND NONDETERMINISTIC AUTOMATA

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 83
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 83
Espoo 2003 HUT-TCS-A83

ON TRANSLATING LINEAR TEMPORAL LOGIC INTO ALTERNATING AND NONDETERMINISTIC AUTOMATA

Heikki Tauriainen

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
FIN-02015 HUT
Tel. +358-0-451 1
Fax. +358-0-451 3369
E-mail: lab@tcs.hut.fi

ISBN 951-22-6868-X
ISSN 1457-7615

Multiprint Oy
Helsinki 2003

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

This thesis develops the theory of translating future-time propositional linear temporal logic (LTL) into nondeterministic automata via linear alternating automata. Unlike nondeterministic automata, linear 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 proposed unified generalized definition for both alternating and nondeterministic automata facilitates combining the rules with new minimization heuristics for alternating automata, based on the general theory of using language containment tests and structural analysis of automata for removing transitions. The generalized definition supports translation of linear alternating automata into nondeterministic automata within the best known limit for the worst case increase in the number of states of the resulting automata. The translation construction also reveals a simple syntactic subclass of LTL for which the exponential increase in the number of states can always be avoided. Additionally, the emptiness of generalized nondeterministic automata is shown to be decidable without degeneralization by using a new variant of the well-known nested depth-first search algorithm.

Keywords: linear temporal logic, alternating automata, automata minimization, nested depth-first search

CONTENTS

1 INTRODUCTION

Automata on infinite objects link the theory of reasoning about the correctness of finite-state systems to the design of concrete decision procedures for checking the satisfiability of formal logical specifications and for the automatic verification of systems against these specifications [11, 45, 56] (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. Whether the system satisfies its specification can then be decided by checking whether none of the computations of the system is accepted by a finite automaton that encodes all "incorrect" computations as a result of having been generated from the logical specification using a *translation procedure*. The connection between automata and logic arises from the classic interpretation of automata as generators of *languages* that consist of sequences (words) or trees satisfying given properties: this connection has led to the introduction of a wide variety of automata designed for capturing the expressive power of many linear- and branching-time logics (e.g., [18, 40, 42, 50, 74, 75, 76]).

In practice, the extreme sensitivity of system descriptions to rapid combinatorial explosion (the *state space explosion problem*—see, for example, the survey by Valmari [71]) presents difficult challenges for the efficient automatic analysis of systems. This same problem plagues also the translation of specifications into automata, and its severity depends on the expressiveness of the chosen specification logic: the combinatorial explosion can range from polynomial to nonelementary complexity in the original size of the specification for logics that are nevertheless known to be decidable (see, for example, Thomas [70]). The struggle against the combinatorial explosion thus presents a need for practically efficient translation procedures between logics and automata.

This work focuses on automata translation procedures for specifications given as formulas of classic future-time *propositional linear temporal logic* (LTL), which supports reasoning about the properties of (nonbranching) computation paths in a system using invariants, future-time assertions, and qualitative causality and fairness constraints on the occurrence of states in the computations [26, 54, 55]. Procedures for translating LTL into automata have been studied extensively in the literature. All of these constructions translate LTL either into *nondeterministic* (see, for example, Thomas [69]) or *alternating* [6, 9, 49, 52] finite automata on infinite words. Whether an infinite input word is accepted by an automaton is then determined, e.g., by a set of designated states that the automaton should visit infinitely often when processing the word (a condition commonly known as *Büchi acceptance*).

Alternation generalizes nondeterminism (existential choice for the "next" state of the automaton) by combining it with universal choice to allow the automata to move to possibly several states at once. As opposed to constructions for translating LTL directly into nondeterministic automata [14, 15, 30, 31, 32, 43, 59, 60, 62, 68, 75, 78], the size of which may be exponential in the length of the LTL specifications, the opportunity to mix existential

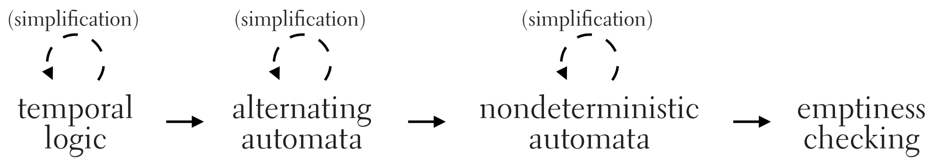


Fig. 1.1: The verification procedure for LTL as a series of translations

and universal choice in automata leads to translation constructions, the results of which are only linear in the size of the specifications [24, 28, 29, 37, 48, 58, 72]. This seemingly obvious advantage of using alternating automata in place of nondeterministic automata to avoid combinatorial explosion comes with a cost, however, since the extra succinctness in representation prevents working with alternating automata using the same algorithmic techniques that apply to nondeterministic automata. This difficulty is traditionally overcome by replacing alternation with nondeterminism using one of the *nondeterminization constructions* proposed in the literature for alternating automata [18, 24, 28, 37, 49, 51, 58]. Even though the exponential worst-case combinatorial cost of these constructions seemingly voids any advantages of the effort of translating LTL into alternating automata in the first place, alternating automata have nevertheless been argued to provide useful additional insight into the translation procedure [73]. Namely, direct translation of LTL into nondeterministic automata misses a two-way correspondence between the expressiveness of LTL and automata: while nondeterministic automata are in general strictly more expressive than LTL [77], LTL has been shown [47, 58] to be expressively equivalent to a simple subclass of alternating automata known as *very weak* [37, 58] or *linear* [47] alternating automata. This correspondence between logic and automata may then allow special simplification and nondeterminization constructions that can be used to improve the overall translation with optimizations that are less obvious (or even impossible in general) for the strictly more expressive nondeterministic automata. Of course, the succinctness of alternating automata also hints at the (admittedly very ambitious) possibility of avoiding the explicit use of nondeterministic automata by discovery of techniques for the efficient direct manipulation and analysis of alternating automata.

Automata-theoretic verification of linear temporal logic specifications can be seen as a series of translations as shown in Fig. 1.1. The last step corresponds to checking the set of computations of a given system that violate a given temporal logic specification (obtained by combining the nondeterministic automaton with the system) for *emptiness*. The same technique applies to testing the satisfiability of the logical specification itself. To counter combinatorial explosion, many improvements have been proposed in the literature for the minimization of intermediate results. Common optimizations include rewriting rules for LTL specifications (e.g., [21, 62, 68]), minimization techniques for alternating [25, 28, 58] and nondeterministic automata [20, 21, 22, 34, 62, 68], as well as special *on-the-fly* translation and model checking algorithms [13, 14] that allow interleaving all phases of the translation together by constructing the automaton from the specification on demand during emptiness checking.

In practice, the implementation of the verification procedure involves many decisions, such as choosing definitions to be used for the underlying automata. To maximize the efficiency of the 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. Actually, 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 variations in the interpretation of acceptance, which can be specified also in terms of a designated set of *transitions* instead of a set of states [14, 28, 32, 68]. Although this simple change in the notion of acceptance does not add to the expressiveness of the automata, the state- and transition-based interpretations are nevertheless asymmetric when the size of the representation is considered. Namely, even though state-based acceptance can easily be reduced to transition-based acceptance without increasing the size of the automata (visiting a state infinitely often implies taking a transition leaving the state infinitely often), the converse reduction may require adding new states to the automata in the general case (see, for example, Giannakopoulou and Lerda [32]). It can thus be said that transition-based acceptance generalizes state-based acceptance. A similar asymmetry concerns the placement of labels (i.e., the symbols that the automaton reads from its input) in the automata; again, placing the labels on transitions instead of states leads to more succinct definitions for automata.

Acceptance in automata can be generalized also in other ways. For example, in most constructions for translating LTL into nondeterministic automata, the set of accepting states is replaced with 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* [31] was introduced mainly as a conceptual aid for describing translation procedures. Similar to the asymmetry between state- and transition-based acceptance, however, automata with multiple sets of accepting states are more succinct than automata with a single set of accepting states [66], despite the expressive equivalence of the automata (established by replacing generalized acceptance with non-generalized acceptance by "unfolding" the automata [10, 13, 19, 28, 31, 32]). Although still more succinct representations for automata can be obtained through further generalizations of acceptance with no change in the expressiveness of the automata (see, for example, Thomas [70]), they have not been widely used in the context of translating LTL into automata.

Because of the above reasons, automata with families of sets of "accepting" transitions have been proposed as a basis for an efficient implementation of the LTL verification procedure [14, 28, 32, 68]. To take the best possible advantage of these automata, each verification step should clearly 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, few translation constructions (with the exception of the one proposed by Couvreur [14]) strive to achieve this goal fully in practice: most other translations rely on additional conversions between formalisms to finally obtain automata with a classic Büchi acceptance condition specified using a set of states. This work presents a unified approach to LTL translation via alternating automata with accepting transitions and generalized acceptance.

Related Work

The connection between logics and automata on infinite objects was first used in the 1960s by Büchi [7] and Rabin [57] to prove decidability results for monadic second-order logics with successor functions (see, for example, Thomas [69], for a survey of the classic results). In the 1980s, these automata were used for obtaining decision procedures for many modal and temporal logics of programs. Streett [64] applied automata to the decision problem of extended propositional dynamic logic; Vardi, Wolper and Sistla [76, 80] used automata-theoretic techniques for deciding extended linear-time temporal logics; and Emerson and Sistla [19] proposed an automata-theoretic decision procedure for the full branching-time temporal logic CTL*. As a special case of their own construction [76, 80], Vardi and Wolper proposed also an explicit decision and model checking procedure for LTL [75]. In 1995, Gerth, Peled, Vardi and Wolper [31] presented an on-the-fly tableau construction for translating future-time LTL into nondeterministic automata with generalized acceptance. Unlike previous constructions, which usually defined automata as tableaux of the worst-case exponential size, the construction of Gerth et al. was an explicit procedure for building only the actually relevant part of an automaton. This advantage of the construction made it a popular source of many related translation procedures, from direct improvements to the explicit tableau construction [15, 32, 60, 62, 68, 78] to translations that support a symbolic representation of the automata [14, 59]. Actually, many symbolic tableau methods used as an alternative source of decision and model checking procedures for LTL [8, 12, 38, 39, 45] can be seen as another line of automata translation procedures via a straightforward interpretation of the tableaux as nondeterministic automata. These procedures, similar to the early automata constructions [19, 75], do not focus on issues such as the minimization of automata, however.

The notion of using automata with accepting transitions for LTL translation was proposed by Couvreur [14] and later advocated also by Gastin and Oddoux [28], Thirioux [68], and Giannakopoulou and Lerda [32].

Muller, Saoudi and Schupp [50] demonstrated the connection between special classes of alternating automata and temporal logics by giving a translation from a branching-time version of the extended temporal logic of Wolper [77] to *weak* alternating automata [51]. Explicit constructions for LTL were later presented by Vardi [72] and Isli [37]; Rohde [58] worked with linear temporal logics on transfinite sequences. Miyano and Hayashi [49] gave the first nondeterminization construction for alternating automata on infinite words; related constructions for more general tree automata were later given by Muller, Saoudi and Schupp [51], and Emerson and Jutla [18], and for automata on transfinite words by Rohde [58]. Optimizations to the nondeterminization of word automata have been further studied by Isli [37], Gastin and Oddoux [28] and Fritz [24]. In particular, by combining nondeterminization with multiple sets of accepting transitions in the nondeterministic automata, Gastin and Oddoux were able to obtain an implementation that was very competitive against implementations based on direct translation of LTL into nondeterministic automata [28].

Translating LTL into automata has also raised interest in general techniques for the minimization of automata. In addition to minimization techniques that exploit special structural properties of automata (e.g., [21, 28, 58, 62, 68]), minimization constructions based on various *simulation relations* have also been proposed for both nondeterministic [20, 21, 22, 34, 62] and alternating automata [24, 25].

Although most translations—including the one presented in this work—from LTL into automata concentrate only on future time, constructions for LTL with past time connectives and temporal logics with two-way automata operators have also been presented for both nondeterministic [59, 75] and alternating automata [29, 40, 48]. Extensions to more expressive logics include constructions for the propositional μ-calculus [18], and automata translation procedures for temporal logics on infinite *traces* instead of words [1, 27]. On the other hand, special constructions targeted towards efficient automata translation of the safety fragment of LTL have also been proposed [30, 43].

Contributions and Organization of This Work

This work presents a unified approach to translating future-time LTL into automata and checking the emptiness of the automata, using transition-based generalized acceptance for the automata throughout all constructions. Instead of moving from LTL directly to nondeterministic automata [14], however, the construction proceeds via translation of LTL first into an expressively equivalent restricted subclass of alternating automata called *linear* or *very weak alternating automata* [37, 47, 58]. The special properties of these automata are reviewed together with the definitions of linear temporal logic and generalized alternating automata in Ch. 2. Throughout the work, all constructions used in the transformation of automata are given direct low-level correctness proofs using a basic toolset of simple results on the behavior of generalized alternating automata. Also this toolset is laid out in Ch. 2.

Similar to classic constructions for building nondeterministic finite word automata from regular expressions, the procedure for translating linear temporal logic into linear alternating automata (Ch. 3) has an intuitive description as a sequence of steps of joining automata incrementally into more complex automata using simple *translation rules* that capture the semantics of the underlying logic. The basic construction is closely related to the one given by Gastin and Oddoux [28] and satisfies the best known upper bounds for the size of automata corresponding to LTL formulas [14, 28]. However, the generalized definition for linear alternating automata, despite its expressive equivalence with LTL (Ch. 3), gives rise to new heuristics for the simplification of automata during the incremental translation procedure. These heuristics, discussed in Ch. 4, include new translation rules for syntactic special cases, as well as local structural automata minimization techniques, the implementation of which benefits also from the restricted structure of the automata. In general, opportunities for simplification are implied by special cases of *language containment relationships* between alternating automata.

The translation construction reveals a syntactic subclass of LTL that supports translation into nondeterministic automata without exponential combinatorial blow-up (Ch. 5). Nondeterminization of generalized linear alternating automata with accepting transitions is shown to be (still) possible in the general case with an exponential increase in the number of states in the automata; in the general case, however, the construction also requires increasing the number of generalized acceptance conditions. Finally, the emptiness of nondeterministic automata with generalized acceptance is shown in Ch. 6 to be decidable directly using a generalized version of the classic *nested depth-first search* emptiness checking algorithm by Courcoubetis, Vardi, Wolper and Yannakakis [13], an efficient on-the-fly emptiness checking algorithm for classic Büchi automata.

2 DEFINITIONS AND BASIC RESULTS

2.1 MATHEMATICAL CONCEPTS AND NOTATION

We assume basic knowledge on sets, ordered tuples, relations and functions (mappings), and the principle of mathematical induction. We shall work with the set of natural numbers $\mathbb{N} \stackrel{def}{=} \{0, 1, 2, \ldots\}$ extended with an element $\omega \notin \mathbb{N}$ (the first infinite ordinal). We assume the elements of $\mathbb{N}$ to be ordered into their traditional linear order by the (irreflexive and transitive) relation $< \subseteq \mathbb{N} \times \mathbb{N}$ with its usual semantics. For the applications in this work, we extend the relation $<$ to $\big(\mathbb{N} \cup \{\omega\}\big) \times \big(\mathbb{N} \cup \{\omega\}\big)$ by considering ω an element that satisfies $\omega \not< \omega$ and $n < \omega$ for all $n \in \mathbb{N}$. Likewise, we extend the traditional addition operation $+$ in $\mathbb{N}$ to the set $\mathbb{N} \cup \{\omega\}$ by defining $\omega + n \stackrel{def}{=} n + \omega \stackrel{def}{=} \omega + \omega \stackrel{def}{=} \omega$. For convenience, comparison between elements of $\mathbb{N} \cup \{\omega\}$ is often denoted also by the operators $=$, $\leq$, $\geq$ and $>$ with their usual semantics.

If X is a countable set, we denote by $|X|$ the cardinality of X, also called the *size* of X. Two countable sets X and Y are *isomorphic* if and only if (iff) they have the same cardinality (equivalently, iff there exists a bijective mapping $f : X \to Y$). A set is *(countably) infinite* iff it is isomorphic to the set of natural numbers $\mathbb{N}$. If X is countably infinite, we define $|X| \stackrel{def}{=} \omega$. If $|X| < \omega$, we define the *powerset* of X as the set that contains all subsets of X and denote it by 2^X.

If X is a subset of another set Y, we call the set $Y \setminus X$ the *complement of X with respect to Y*. When the set Y is clear from the context, we 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* $I = \{n \in \mathbb{N} \mid n < m \text{ for some } 0 \leq m \leq \omega\}$ to X. For all $i \in I$, $x(i)$ is called the i^{th} *element of x*. We may also describe x by "listing its elements" as $x = (x_i)_{0 \leq i < |x|} = (x_0, x_1, x_2, \ldots)$, where $x_i \stackrel{def}{=} x(i)$. 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 (countably) 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 $i \leq j \leq |I|$. 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 $\big(x(k)\big)_{i \leq k < j}$. 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 $\big(x(k)\big)_{i \leq k < j}$ using the simpler notation x^i. Clearly, the suffix x^i is infinite iff x is infinite.

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 $x : \{n \in \mathbb{N} \,|\, n < |I_1| + |I_2|\} \to X_1 \cup X_2$ defined by

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

Because concatenation is an associative operation, i.e., $(xy)z = x(yz)$ 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. 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 the elements of X, parentheses "(" and ")" and the symbols $\cup$, * and $^\omega$ 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 *regular expressions* over X):

- Each element of X is a regular expression.
- If α and β are regular expressions, then $(\alpha \cup \beta)$, $(\alpha\beta)$ and α^* are regular expressions.
- If α is a regular 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 ω-regular expression α defines a set of words over X. We denote the set of words defined by the ω-regular expression α by $\mathcal{L}(\alpha)$ and call this set the *language* of α. Formally, $\mathcal{L}(\alpha)$ is defined for regular and ω-regular expressions as follows:

- $\mathcal{L}(\alpha) \stackrel{def}{=} \{x : \{0\} \to X \,|\, x(0) = \alpha\}$ 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 regular or both ω-regular expressions (the sequences that belong to either or both of $\mathcal{L}(\alpha)$ and $\mathcal{L}(\beta)$);
- $\mathcal{L}\big((\alpha\beta)\big) \stackrel{def}{=} \{xy \,|\, x \in \mathcal{L}(\alpha), y \in \mathcal{L}(\beta)\}$ for any regular expression α and any regular or ω-regular expression β (the sequences formed by concatenating a sequence from $\mathcal{L}(\beta)$ to a sequence in $\mathcal{L}(\alpha)$);
- $\mathcal{L}(\alpha^*) \stackrel{def}{=} \{\varepsilon_X\} \cup \bigcup_{1 \le i < \omega} \{x_1x_2 \dots x_i \,|\, x_1, x_2, \dots, x_i \in \mathcal{L}(\alpha)\}$ for any regular expression α (the set of sequences obtained by finite concatenations of zero or more sequences in $\mathcal{L}(\alpha)$)

- $\mathcal{L}(\alpha^\omega) \stackrel{def}{=} \{x_1x_2x_3\ldots \mid x_i \in \mathcal{L}(\alpha) \setminus \{\varepsilon_X\} \text{ for all } 1 \leq i < \omega\}$ for any regular 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". In addition, we simplify the notation by omitting parentheses from the 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 denotes the infinite word that consists of the symbol a, $a^\omega \cup b^*c^\omega$ represents all infinite words consisting of either the symbol a or a 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 TEMPORAL LOGIC

As shown in the example at the end of the previous section, ω-regular expressions provide a means for specifying simple properties of 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, given two ω-regular expressions α and β, it is not obvious how (or whether) these expressions could be used to form another ω-regular expression corresponding to the language $\mathcal{L}(\alpha) \cap \mathcal{L}(\beta)$, or an ω-regular expression that represents all words that do *not* belong to $\mathcal{L}(\alpha)$. Even though these languages are in fact expressible using ω-regular expressions [7], reasoning about the properties of computation paths of finite-state systems is often more convenient using logical operations for transforming simple expressions into more complex expressions. In this chapter we review the syntax and semantics of classic future-time *propositional linear temporal logic* [26, 54, 55]; despite the difference in the basic operations for manipulating expressions in the logic, the languages definable using LTL actually form a strict subset of the languages definable using ω-regular expressions (see, for example, Thomas [69]).

2.2.1 Syntax

Let AP be a finite set of atomic propositions. The set $LTL(AP)$ of propositional future-time linear 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 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 subformulas of φ is denoted by $\mathrm{Sub}(\varphi)$. Identical subsequences of φ are identified as the same subformula; thus, $|\mathrm{Sub}(\varphi)|$ equals the number of syntactically distinct subformulas in φ.

The formula $\varphi \in LTL(AP)$ is called a *literal* iff $\varphi = p$ or $\varphi = \neg p$ for some atomic proposition $p \in AP$; literals and the symbol $\top$ are called *atomic* subformulas of φ. If φ is not a literal (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*, where $\circ$ is the *main connective* of φ, 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 $\varphi = (\varphi_1 \, \mathsf{U_s} \, \varphi_2)$ for some $\varphi_1, \varphi_2 \in LTL(AP)$. We denote by $\mathrm{Temp}(\varphi)$ the maximal subset of $\mathrm{Sub}(\varphi)$ consisting of only pure temporal formulas.

2.2.2 Semantics

Basic operators

Linear 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 classic semantics of linear temporal logic is defined using a binary relation $\models$ between infinite sequences $w \in (2^{AP})^\omega$ of subsets of AP and formulas of the logic inductively 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 $\mathcal{L}(\varphi) \stackrel{def}{=} \{w \in (2^{AP})^\omega \,|\, w \models \varphi\}$ of all models of φ is called the *language* of φ. The formula φ is *satisfiable* if $\mathcal{L}(\varphi) \neq \emptyset$ 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* $\overline{\mathcal{L}(\varphi_1)} \stackrel{def}{=} (2^{AP})^\omega \setminus \mathcal{L}(\varphi_1)$ of the language of φ with respect to $(2^{AP})^\omega$ equals $\mathcal{L}(\neg\varphi_1)$. For a pair of formulas $\varphi, \psi \in LTL(AP)$, we write $\varphi \equiv \psi$ to denote that

$\mathcal{L}(\varphi) = \mathcal{L}(\psi)$. If ψ is a subformula of φ and ψ' is another LTL formula such that $\psi \equiv \psi'$, then any single occurrence of ψ in φ can be substituted with ψ' without changing the language of φ.

If $\varphi \in PL(AP)$, we project the satisfiability 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 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 altering the semantics of the logic. Standard extensions include the Boolean constant $\perp$ ("false"), the propositional connectives $\wedge$ (conjunction), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence) and $\oplus$ (exclusive disjunction) as well as temporal connectives such as

- 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 shall also use the common shorthand notation $\bigvee_{\varphi \in \Phi} \varphi$ and $\bigwedge_{\varphi \in \Phi} \varphi$ to denote an (arbitrarily parenthesized) LTL formula formed by joining the elements of a given set of formulas $\Phi \subseteq LTL(AP)$ (in any order) with either the $\vee$ or the $\wedge$ connective, respectively. (The parenthesization and the order of the formulas do not matter due to the associativity and commutativity of these connectives.) It is customary to define $\bigvee_{\varphi \in \emptyset} \varphi \overset{def}{\equiv} \perp$ and $\bigwedge_{\varphi \in \emptyset} \varphi \overset{def}{\equiv} \top$.

In this work, we assume all LTL formulas to be written using atomic propositions, Boolean constants $\top$ and $\perp$, 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 of the U and R connectives will sometimes be omitted if the strength of the connective is not relevant. A formula having one of these connectives as its main connective is called a (strong or weak) *temporal eventuality*.

The models of the temporal eventualities are infinite sequences over 2^{AP} that may or must 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 holds throughout the entire sequence.

(In our notation, $\mathsf{U_s}$ and $\mathsf{R_w}$ correspond to the traditional Until and Release connectives commonly used in the literature.)

2.2.3 Positive Normal Form

Using the operator set fixed above, any formula $\varphi \in LTL(AP)$ can be written as an equivalent LTL formula φ' ($\varphi \equiv \varphi'$) in which all negations precede atomic propositions. The positive normal form of a formula φ can be found by applying the following well-known identities

$$\begin{aligned}
\neg\top &\equiv \bot & \neg\bot &\equiv \top \\
\neg\neg\varphi &\equiv \varphi & & \\
\neg(\varphi \vee \psi) &\equiv (\neg\varphi \wedge \neg\psi) & \neg(\varphi \wedge \psi) &\equiv (\neg\varphi \vee \neg\psi) \\
\neg\mathsf{X}\varphi &\equiv \mathsf{X}\neg\varphi & & \\
\neg(\varphi\,\mathsf{U_s}\,\psi) &\equiv (\neg\varphi\,\mathsf{R_w}\,\neg\psi) & \neg(\varphi\,\mathsf{R_w}\,\psi) &\equiv (\neg\varphi\,\mathsf{U_s}\,\neg\psi) \\
\neg(\varphi\,\mathsf{U_w}\,\psi) &\equiv (\neg\varphi\,\mathsf{R_s}\,\neg\psi) & \neg(\varphi\,\mathsf{R_s}\,\psi) &\equiv (\neg\varphi\,\mathsf{U_w}\,\neg\psi)
\end{aligned}$$

such that each subformula of φ matching the left-hand side formula of some identity is replaced with the right-hand side formula until none of the subformulas of the resulting formula φ' matches the left-hand side of any identity. Since each identity replaces a negated subformula with a pure temporal formula iff the subformula itself is a pure temporal subformula, the number of pure temporal subformulas cannot increase in the conversion. (It is possible, however, that $|\mathrm{Temp}(\varphi')| < |\mathrm{Temp}(\varphi)|$, for example, if $\varphi = (\mathsf{X}\neg p \wedge \neg\mathsf{X}p)$.) In this work, all LTL formulas are assumed to be in positive normal form unless otherwise noted.

Let $\varphi \in LTL(AP)$ be an LTL formula in positive normal form. The *size* of φ (denoted $|\varphi|$) is defined as the total number of atomic and compound subformulas in φ, where all identical subformulas of φ contribute to the result (that is, we do not identify syntactically identical subformulas here). Formally, we define the size of φ inductively as

$$|\varphi| \stackrel{def}{=} \begin{cases} 1 & \text{if } \varphi \in \{\top, \bot, p, \neg p\},\, p \in AP \\ 1 + |\varphi_1| & \text{if } \varphi = \mathsf{X}\varphi_1 \\ 1 + |\varphi_1| + |\varphi_2| & \text{if } \varphi = (\varphi_1 \circ \varphi_2),\, \circ \in \{\vee, \wedge, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\} \end{cases}$$

where $\varphi_1, \varphi_2 \in LTL(AP)$.

2.3 ALTERNATING AUTOMATA

The possibility of mixing nondeterministic (existential) behavior with universal behavior (corresponding, in a sense, to parallelism) in finite automata working on finite words was first investigated by Chandra, Kozen and Stockmeyer [9], who defined the notion of *alternating automata*, and Brzozowski and Leiss [6], who introduced an equivalent concept of *Boolean automata*. Alternation was later studied within the context of infinite inputs by Miyano and Hayashi [49], and Muller and Schupp [52], who considered alternating automata on infinite words and trees, respectively. Unlike a nondeterministic automaton that always chooses a single "next state" at each step when working on its input, an alternating automaton can choose several of these "next states" at once, spawning independent copies of itself that work independently on the remaining input. Whether the automaton accepts its input is then determined by a condition that specifies which of the spawned copies should accept or reject the remainder of the input.

In the case of finite inputs, alternation allows for a representation of automata that does not add to the expressiveness of plain nondeterministic automata, but is in the best case only logarithmic in the size of the smallest possible corresponding nondeterministic representation [6, 9, 44]. Conversely, every alternating automaton on finite words can be translated into an equivalent nondeterministic automaton with a worst-case exponential blow-up in the number of states; an analogous result holds for commonly used classes of automata on infinite inputs [49]. Together with the wide expressiveness of automata in general, the succinctness of alternating automata provides the main motivation for the study of using them in the automatic verification of systems against their specifications. In this section, we review the basic definitions and properties of alternating automata on infinite words and some of the subclasses of the automata.

2.3.1 Basic Properties

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 on the states of the automata [6, 9]; however, it is common to restrict the use of negation when working with automata on infinite inputs [52, 72]. While the Boolean representation is convenient for proving several fundamental properties of alternating automata, such as a complementation construction based on syntactic manipulation of the Boolean functions [52], the notion of nondeterministic choice between individual transitions of the automata is not explicit in the Boolean representation. Such a notion is nevertheless useful, for example, for representing the automata graphically using common drawing techniques borrowed from nondeterministic automata. As we shall see later in Ch. 4, 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 [28]; in this definition, the explicit representation of the individual transitions leaving the same state is equivalent to the disjunctive normal form[1] of a corresponding Boolean encoding that does not include any negations.

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*.

The components of each transition $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ ($q \in Q$, $\Gamma \subseteq \Sigma, F \subseteq \mathcal{F}, Q' \subseteq Q$) are called the *start state*, the *guard*, the *acceptance conditions* and the *target states* of t, respectively. A transition is an *initial transition* of $\mathcal{A}$ iff its start state is q_I, and it is a *self-loop* iff it includes its start

[1] That is, a formula of the form $\bigvee_{0 \le i \le n} \varphi_i$, where $\varphi_i \equiv \bigwedge_{1 \le j \le m_i} \psi_{i,j}$, and every $\psi_{i,j}$ is an atomic formula ($n, m_i \in \mathbb{N}$).

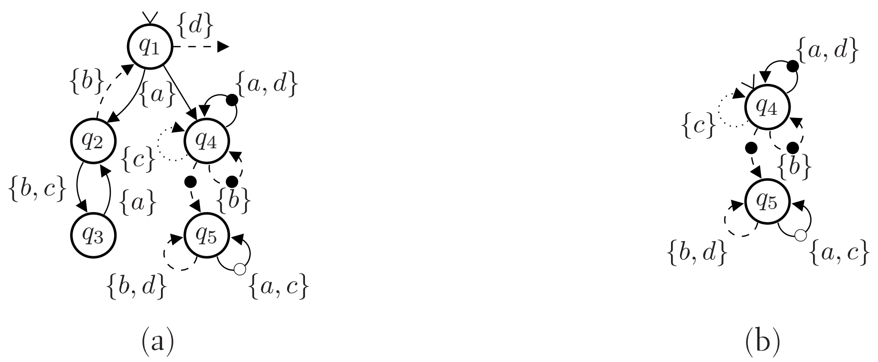


Fig. 2.1: (a) Graphical representation of the alternating automaton $\mathcal{A} \stackrel{def}{=} \big\langle \{a,b,c,d\}, \{q_1,q_2,q_3,q_4,q_5\}, \big\{\langle q_1,\{a\},\emptyset,\{q_2,q_4\}\rangle, \langle q_1,\{d\},\emptyset,\emptyset\rangle, \langle q_2,\{b\},\emptyset,\{q_1\}\rangle, \langle q_2,\{b,c\},\emptyset,\{q_3\}\rangle, \langle q_3,\{a\},\emptyset,\{q_2\}\rangle, \langle q_4,\{a,d\},\{f_1\},\{q_4\}\rangle, \langle q_4,\{b\},\{f_1\},\{q_4,q_5\}\rangle, \langle q_4,\{c\},\emptyset,\{q_4\}\rangle, \langle q_5,\{a,c\},\{f_2\},\{q_5\}\rangle, \langle q_5,\{b,d\},\emptyset,\{q_5\}\rangle\big\}, q_1, \{f_1,f_2\}\big\rangle$;
(b) The subautomaton $\mathcal{A}^{q_4} = \big\langle \{a,b,c,d\}, \{q_4,q_5\}, \big\{\langle q_4,\{a,d\},\{f_1\},\{q_4\}\rangle, \langle q_4, \{b\},\{f_1\},\{q_4,q_5\}\rangle, \langle q_4,\{c\},\emptyset,\{q_4\}\rangle, \langle q_5,\{a,c\},\{f_2\},\{q_5\}\rangle, \langle q_5,\{b,d\},\emptyset,\{q_5\}\rangle\big\}, q_4, \{f_1,f_2\}\big\rangle$

state in its target states.

The *size* $|\mathcal{A}|$ of the automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ is defined to be equal to the number of states in the automaton.

Example 2.3.1 We illustrate our conventions for drawing alternating automata in Fig. 2.1. The states of the automata are denoted by 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 between 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 $|Q'| = n$ for some $0 \leq n < \omega$, we draw $\max\{1, n\}$ arrows from the state q to each state in Q' (thus, if $Q' = \emptyset$, we draw a single arrow that is not connected with any other state). Arrows associated with the same transition are drawn in the same line style; since each transition has a unique start state, the same line styles can be reused in each state of the automaton without ambiguity. Acceptance conditions in F are represented by small shaded circles on the transition arrows, where each different shade corresponds to a different acceptance condition. For simplifying 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 associated with 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 $q \in Q$. If there exists a transition $\langle q, \Gamma, F, Q'\rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$, then each state $q' \in Q'$ is called a *successor* of q. A *path* in $\mathcal{A}$ is a sequence $x = (q_i)_{0 \leq i < n+1} \in Q^{n+1}$, where $0 \leq n \leq \omega$ and q_{i+1} is a successor of q_i for all $0 \leq i < n$. If $n = 0$, the path is *trivial*; if $n < \omega$, the path is a *finite* path *from* q_0 *to* q_n; 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 $0 \leq i < n+1$. A path is *simple* iff $q_i \neq q_j$ holds for all $0 \leq i, j < n+1$, $i \neq j$, and it is a *loop* (alternatively, a *cycle*) iff $1 \leq n < \omega$, $q_n = q_0$ and $q_i \neq q_j$ for all $0 \leq i, j \leq n-1$, $i \neq j$. 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 $\mathcal{A}$.

Example 2.3.2 In Fig. 2.1 (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. ■

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, $\mathcal{A}^q = \langle \Sigma, Q^q, \Delta^q, q_I^q, \mathcal{F}^q \rangle$, where $Q^q \stackrel{def}{=} \{q\} \cup \{q' \in Q \mid q' \text{ is a descendant of } q\}$, $\Delta^q \stackrel{def}{=} \big\{\langle q', \Gamma, F, Q' \rangle \in \Delta \,\big|\, q' \in Q^q, \Gamma \subseteq \Sigma, F \subseteq \mathcal{F} \text{ and } Q' \subseteq Q^q \big\}$, $q_I^q \stackrel{def}{=} q$, 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 $\mathcal{A}$.

Example 2.3.3 Figure 2.1 (b) shows the subautomaton $\mathcal{A}^{q_4}$ obtained from the automaton $\mathcal{A}$ depicted in Fig. 2.1 (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 labelled acyclic graph $G = \langle V, E, L \rangle$. Here, V is the set of *nodes* (assumed to be partitioned into disjoint finite *levels* $V_i \subseteq V$ such that $V = \bigcup_{0 \leq i < \omega} V_i$ and $V_i \cap V_j = \emptyset$ for all $i \neq j$), $E \subseteq \bigcup_{0 \leq i < \omega} (V_i \times 2^{V_{i+1}})$ is the set of *(hyper)edges*, and $L : (V \cup E) \to (Q \cup \Delta)$ is the *labelling function*. For convenience, given a set $X \subseteq V \cup E$, we use the shorthand notation $L(X)$ to represent the set of labels $\{L(x) \mid x \in X\}$. In addition, V, E and L must satisfy the following conditions:

- $V_0 = \{v_0\}$, $L(v_0) = q_I$;
- if $v \in V_i$ for some $0 \leq i < \omega$, then v has the unique outgoing edge $e = \langle v, V' \rangle \in E$ for some $V' \subseteq V_{i+1}$, and there exists a transition $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ such that $q = L(v)$, $w(i) \in \Gamma$, $Q' = L(V')$, and $L(e) = t$; we say in this case that the edge labelling is *consistent*;
- if $v' \in V_i$ for some $1 \leq i < \omega$, then there exists a node $v \in V_{i-1}$ and an edge $e = \langle v, V' \rangle \in E$ such that $v' \in V'$.

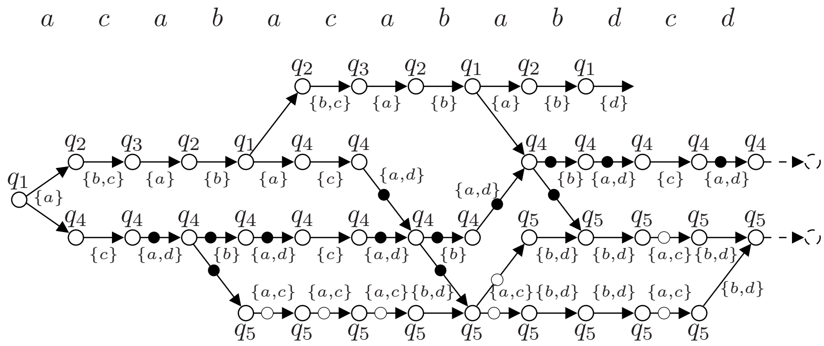


Fig. 2.2: First few levels of a run of the alternating automaton in Fig. 2.1 (a) formed by reading the input $acabacababdcd$

Two edges $e_1 = \langle v_1, V' \rangle \in E$ and $e_2 = \langle v_2, V'' \rangle \in E$ are *consecutive* iff $v_2 \in V'$. Similar to alternating automata, we call v_2 a *successor* of v_1 in this case; descendants of a node and paths in a run are defined analogously.

Example 2.3.4 Figure 2.2 illustrates the construction of the first few levels of (one possible) run for the alternating automaton in Fig. 2.1 (a) on the input $acabacababdcd$. Again, we represent nodes of the run with circles and edges with (sets) of arrows between 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. We represent this by drawing arrows from the node labelled with the state q_1 to two new nodes labelled with these states in the figure to form level V_1 of the run. Nodes belonging to the same level are always drawn horizontally aligned (with their labels shown beside the nodes themselves). The following levels of the run of the automaton are defined in a similar manner such that the labels of the successors of each node always coincide with the target states of some transition that starts from the state labelling the node itself and includes the next input symbol (shown at the top of the figure) in its guard.

The number of nodes in a level of the run corresponds to the number of currently active copies of the automaton; as seen from 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 (caused by the automaton spawning new copies of itself, by a copy taking a transition with an empty set of target states, or even by several copies "merging together").

The arrows leaving each node constitute the unique (hyper)edge starting from the node. Formally, each of these edges is labelled 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. ∎

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 $w \in \Sigma^\omega$. We define the set $\mathcal{E}(G)$ of *acceptance* *sequences* of G as the set of all infinite sequences of consecutive edges of E beginning with the unique edge $\langle v_0, V_1\rangle \in E$; formally,

$$\mathcal{E}(G) \stackrel{\text{def}}{=} \Big\{(e_i)_{0\leq i<\omega} \;\Big|\; \forall 0 \leq i < \omega : e_i \in E \cap (V_i \times 2^{V_{i+1}}), \\ e_i \text{ and } e_{i+1} \text{ consecutive}\Big\}.$$

For a nonempty $\mathcal{E}(G)$ and a sequence $r = (e_i)_{0\leq i<\omega} \in \mathcal{E}(G)$, we further define

$$\inf(r) \stackrel{\text{def}}{=} \Big\{f \in \mathcal{F} \,\Big|\, \forall i \geq 0 : \exists j > i : L(e_j) = \langle q_j, \Gamma_j, F_j, Q'_j\rangle \in \Delta, f \in F_j\Big\}$$

and

$$\text{fin}(r) \stackrel{\text{def}}{=} \Big\{f \in \mathcal{F} \,\Big|\, \exists i \geq 0 : \forall j > i : L(e_j) = \langle q_j, \Gamma_j, F_j, Q'_j\rangle \in \Delta, f \in F_j\Big\}.$$

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

We say that the run G is

- *inf-accepting* iff $\inf(r) = \mathcal{F}$ for all $r \in \mathcal{E}(G)$ (this is analogous to classic Büchi acceptance generalized to multiple acceptance conditions);
- *fin-accepting* iff $\text{fin}(r) = \emptyset$ for all $r \in \mathcal{E}(G)$.

(If all paths through G are finite, $\mathcal{E}(G) = \emptyset$, and thus both conditions hold trivially.)

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 $\mathcal{L}_{\text{inf}}(\mathcal{A})$ or $\mathcal{L}_{\text{fin}}(\mathcal{A})$, 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 $\mathcal{L} = \mathcal{L}_{\text{inf}}(\mathcal{A})$ or $\mathcal{L} = \mathcal{L}_{\text{fin}}(\mathcal{A})$, respectively. The automaton is *inf- (fin-)empty* iff it inf-(fin-)recognizes the empty language. We call two automata *equivalent* iff they recognize the same language (the acceptance modes of the automata will usually be clear from the context).

Example 2.3.5 Consider again the fragment of a run of the automaton of Example 2.3.1 on the input $acabacababdcd$ as shown in Fig. 2.2. This run fragment ends in a level having two nodes labelled 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. 2.2 into a run ending in, for example, an infinite number of identical levels shown in Fig. 2.3 (a). It is easy to see that this run contains a finite number of acceptance sequences, all of which end in an infinite suffix of identically

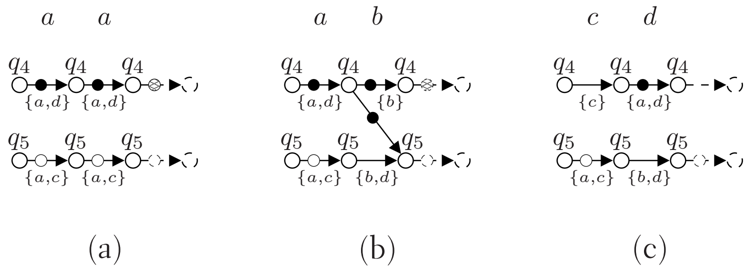


Fig. 2.3: Possible extensions of the graph in Fig. 2.2 into a run of the alternating automaton shown in Fig. 2.1. (a) Extension on the input a^ω; (b) Extension on the input $(ab)^\omega$; (c) Extension on the input $(cd)^\omega$

labelled edges (labelled either with the transition $\langle q_4, \{a,d\}, \{f_1\}, \{q_4\}\rangle$ or the transition $\langle q_5, \{a,c\}, \{f_2\}, \{q_5\}\rangle$). Because neither of these transitions includes both f_1 and f_2 in their respective sets of acceptance conditions, it follows that f_1 and f_2 cannot both occur infinitely often in the labels of edges in all acceptance sequences, and thus the run is not inf-accepting. On the other hand, none of the acceptance sequences satisfies the fin-acceptance condition, either, because both transitions include either f_1 or f_2 in their acceptance conditions.

Figure 2.3 (b) shows another extension for the graph in Fig. 2.2 obtained by concatenating the word $(ab)^\omega$ with the original input. The run will now contain infinitely many acceptance sequences: when choosing a sequence of consecutive edges, we have, in effect, infinitely many opportunities to decide whether to continue a sequence currently ending with an edge corresponding to a "branching" self-loop starting from the state q_4 with an edge labelled with a self-loop starting from q_4 or q_5. Nevertheless, it is easy to see that all acceptance sequences again end in an infinite suffix of edges labelled with self-loops starting from a fixed state of the automaton. More precisely, the edge labels will eventually alternate between the transitions $\langle q_4, \{a,d\}, \{f_1\}, \{q_4\}\rangle$ and $\langle q_4, \{b\}, \{f_1\}, \{q_4, q_5\}\rangle$, or the transitions $\langle q_5, \{a,c\}, \{f_2\}, \{q_5\}\rangle$ and $\langle q_5, \{b,d\}, \emptyset, \{q_5\}\rangle$ in every acceptance sequence. Similar to above, no acceptance sequence satisfies the inf-acceptance condition. However, all sequences ending in a suffix labelled with self-loops starting from q_5 will now satisfy the fin-acceptance condition, because these sequences include infinitely many edges labelled with a transition having an empty set of acceptance conditions. The run is nevertheless not fin-accepting, because the acceptance condition f_1 (the small black circle) will eventually repeat indefinitely in the edge labels of any acceptance sequence ending in edges corresponding to self-loops that start from the state q_4.

Finally, extending the graph in Fig. 2.2 into a run by reading the input $(cd)^\omega$ can be done as shown in Fig. 2.3 (c). As above, the run is not inf-accepting; however, in this case all acceptance sequences contain infinitely many edges labelled with transitions having no acceptance conditions as seen in the figure. Therefore, we see that the automaton fin-accepts the word $acabacababdcd(cd)^\omega$. ∎

Interreducibility of Acceptance Modes

We note that fin-acceptance can always be reduced to the more commonly used (generalized) *Büchi acceptance*—inf-acceptance in our terminology—and vice versa: it is easy to check that an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ fin- (inf)-accepts the language $\mathcal{L} \subseteq \Sigma^\omega$ iff the automaton obtained from $\mathcal{A}$ by complementing the acceptance conditions of each transition of $\mathcal{A}$ with respect to $\mathcal{F}$ inf- (fin)-accepts the same language. (That is, $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}) = \mathcal{L}_{\mathrm{inf}}(\mathcal{A}')$ for the automaton $\mathcal{A}' = \langle \Sigma, Q, \Delta', q_I, \mathcal{F} \rangle$ having 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 $|\mathcal{F}| = 1$, then fin-acceptance coincides with a condition commonly known as *co-Büchi* acceptance.)

As we shall see in Ch. 3, fin-acceptance provides a convenient way to identify certain "bad" transitions that the automaton is not allowed to take indefinitely along any path of a fin-accepting run when working on an input $w \in \Sigma^\omega$ belonging to the language of the automaton. This resembles the requirements that arise in dealing with models of strong temporal eventualities of LTL: recall that an infinite word over 2^{AP} violates a strong temporal eventuality if some designated LTL property remains unsatisfied in all suffixes of the word.

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.

Properties of Runs

In this section we list several elementary results on the 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. 2.1 (a) and Fig. 2.2.

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 $0 \leq i < \omega$. Then, if $v' \in V$ is a descendant of v in G, then $L(v')$ is a descendant of $L(v)$ in $\mathcal{A}$.*

Proof: Clearly, 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 the unique edge starting from v includes v' in its target nodes, and because G is a run, the consistency of the edge labelling 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 there exists a descendant $v' \in V_{i+j}$ of v and an edge $e = \langle v', V' \rangle \in E$ such that $v'' \in V'$ (E contains edges only between consecutive levels of G). Because the edge labelling 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. $\square$

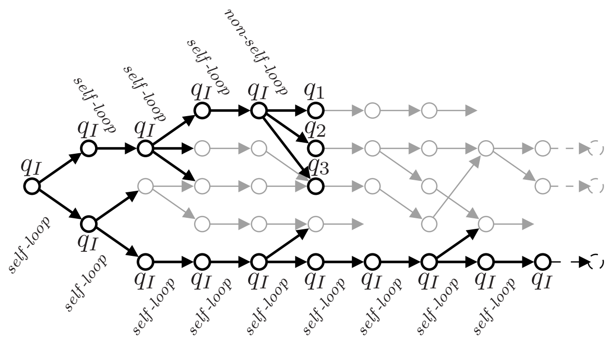


Fig. 2.4: Every run of an alternating automaton contains a sequence of consecutive edges labelled with initial self-loops of the automaton such that the sequence is either infinite, or it is followed by an edge labelled with an initial transition that is not a self-loop

Consider the construction of a run for an alternating automaton $\mathcal{A}$. Clearly, the only way to extend a finite (possibly empty) sequence of consecutive edges labelled with initial self-loops of the automaton with a new edge is to label the new edge with another initial transition of the automaton to ensure the consistency of the labelling. Therefore, every run of the automaton contains either a finite sequence of edges labelled with initial transitions of the automaton such that all edges except the last one correspond to self-loops of the automaton, or an infinite sequence of edges, all of which are labelled with initial self-loops of the automaton (see Fig. 2.4). Because we shall often rely on the existence of such a sequence 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 sequence of consecutive 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 there exists a sequence of consecutive 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 $L(v') = q_I$. Because G is a run, 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 e_{k+1} is labelled 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 $(e_j)_{0 \leq j \leq k+1}$ satisfies the criteria given in the proposition. Otherwise $(e_j)_{0 \leq j \leq k+1}$ is another sequence of consecutive edges labelled with initial self-loops of $\mathcal{A}$. By induction, it follows that we can extract from G a sequence of consecutive edges satisfying the required criteria. $\square$

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. 2.2.

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 $0 \le i < \omega$. Define the graph $G^v = \langle V^v, E^v, L^v\rangle$, where*

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

Let $L(v) = q$. Then 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. It is clear that V^v consists of finite disjoint levels $V^v_j \overset{def}{=} V^v \cap V_{i+j}$ $(0 \le j < \omega)$ such that $E^v \subseteq \bigcup_{0\le j<\omega}(V^v_j \times 2^{V^v_{j+1}})$. By the definitions of G^v and $\mathcal{A}^q$, $V^v_0 = \{v\}$, and $L^v(v) = L(v) = q$ is the initial state of $\mathcal{A}^q$.

Let $v' \in V^v_j \subseteq V_{i+j}$ for some $0 \le j < \omega$. By the definition of G^v, $v' = v$, or v' is a descendant of v in G. Because G is a run of $\mathcal{A}$, there exists a unique edge $e = \langle v', V'\rangle \in E$ $(V' \subseteq V_{i+j+1})$ labelled with a transition $t = \big\langle L(v'), \Gamma, F, L(V')\big\rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w(i+j) \in \Gamma$. It follows that for all $v'' \in V'$, v'' is a descendant of v in G, and (by Proposition 2.3.6) $L(v'')$ is a descendant of q in $\mathcal{A}$. Therefore $V' \subseteq V^v$ and $L(V') \subseteq Q^q$ hold by the definitions of G^v and $\mathcal{A}^q$, which further implies that $e \in E^v$ (and remains unique in E^v) and $t \in \Delta^q$. Additionally, $F \subseteq \mathcal{F}' = \mathcal{F}$ holds by the definition of $\mathcal{A}^q$. Since $L^v(x) = L(x)$ holds for all $x \in V^v \cup E^v$, $t = \big\langle L(v'), \Gamma, F, L(V')\big\rangle = \big\langle L^v(v'), \Gamma, F, L^v(V')\big\rangle$ and $L^v(e) = L(e) = t$, and thus the edge labelling is consistent.

Finally, if $v' \in V^v \setminus \{v\}$, then, by the definition of G^v, v' is a descendant of v in G. Thus, v' is either a successor of v or a successor of a node that is itself a descendant of v in G (and thus belongs to V^v). In either case, there exists an edge $e = \langle v'', V''\rangle \in E$ for some $v'' \in V^v$ and $V'' \subseteq V$ such that $v' \in V''$. Because $v'' \in V^v$, all nodes in V'' are descendants of v in G, and thus $V'' \subseteq V^v$ and $e \in E^v$.

It follows that G^v is a run of $\mathcal{A}^q$ on $\big((w(i+j)\big)_{0\le j<\omega} = \big((w(j)\big)_{i\le j<\omega} = w^i$. $\square$

By focusing only on 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, and let $G = \langle V, E, L\rangle$ be an inf- (fin-)accepting run of $\mathcal{A}$ on $w \in \Sigma^\omega$. Then, for all $0 \le i < \omega$ and $v \in V_i$, the run G^v obtained from G using the construction in Proposition 2.3.8 is an inf- (fin-)accepting run of $\mathcal{A}^{L(v)}$ on w^i. More generally, if $G = \langle V, E, L\rangle$ is an inf- (fin)-accepting run of $\mathcal{A}$ on w, then $\mathcal{A}^{L(v)}$ inf- (fin)-accepts w^i for all $0 \le i < \omega$ 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 the definition of a run and the fact that G^v is a subgraph of G.) It follows that also each acceptance sequence $r^v = (e_j^v)_{0 \leq j < \omega} \in \mathcal{E}(G^v)$ of G^v is a suffix of some acceptance sequence $r = (e_j)_{0 \leq j < \omega} \in \mathcal{E}(G)$ of G (with $e_j^v = e_{i+j}$ for all $0 \leq j < \omega$). Because G is an accepting run of $\mathcal{A}$, r satisfies the inf- (fin-)acceptance condition. Thus, either $\inf(r) = \mathcal{F}$ (inf-acceptance), or $\operatorname{fin}(r) = \emptyset$ (fin-acceptance). Since r^v is an infinite suffix of r, r contains only finitely many edges not contained in r^v, and thus either $\inf(r^v) = \inf(r) = \mathcal{F} = \mathcal{F}^{L(v)}$, or $\operatorname{fin}(r^v) = \operatorname{fin}(r) = \emptyset$. It follows that G^v is an inf- (fin-)accepting run of $\mathcal{A}^{L(v)}$ on w^i. □

Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton, let $w \in \Sigma^\omega$, and let $G = \langle V, E, L \rangle$ be a directed labelled acyclic graph, where, similar to a run of $\mathcal{A}$, V consists of finite disjoint levels V_i ($0 \leq i < \omega$), $E \subseteq \bigcup_{0 \leq i < \omega} (V_i \times 2^{V_{i+1}})$, and $L : (V \cup E) \to (Q \cup \Delta)$. We call G a *partial run* of $\mathcal{A}$ on w iff

- $V_0 = \{v_0\}$, $L(v_0) = q_I$;
- if $v \in V_i$ for some $0 \leq i < \omega$, then either
  - $E \cap \{\langle v, V' \rangle \,|\, V' \subseteq V\} = \emptyset$, that is, v has no outgoing edges, or
  - v has the unique outgoing edge $e = \langle v, V' \rangle \in E$ for some $V' \subseteq V_{i+1}$, and there exists a transition $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ such that $q = L(v)$, $w(i) \in \Gamma$, $Q' = L(V')$ and $L(e) = t$;
- if $v' \in V_i$ for some $1 \leq i < \omega$, then there exists a node $v \in V_{i-1}$ and an edge $e = \langle v, V' \rangle \in E$ such that $v' \in V'$.

Thus, the definition of a partial run is identical to the definition of a run except for the relaxed second condition: a partial run may contain nodes with no outgoing edges. The concept of acceptance sequences extends to partial runs in an obvious way: a partial run G is called a *partial inf- or fin-accepting run* iff each infinite sequence of consecutive edges through G (beginning with the unique edge leaving the node v_0) satisfies the corresponding acceptance condition.

A partial inf- (fin-)accepting run of $\mathcal{A}$ on w can be extended into a complete 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 partial run with no outgoing edges. This fact is formalized in the following proposition; see also Fig. 2.5.

Proposition 2.3.10 *Let $G = \langle V, E, L \rangle$ be a partial inf- (fin-)accepting run of an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ on $w \in \Sigma^\omega$. Let $\widehat{V}_i \overset{def}{=} \{v \in V_i \,|\, E \cap (\{v\} \times 2^{V_{i+1}}) = \emptyset\}$ denote the set of nodes at level i with no outgoing edges ($0 \leq i < \omega$), and assume that $\mathcal{A}^{L(v)}$ has an inf- (fin-)accepting run on w^i for all $0 \leq i < \omega$ and $v \in \widehat{V}_i$. Then, $\mathcal{A}$ inf- (fin-)accepts w.*

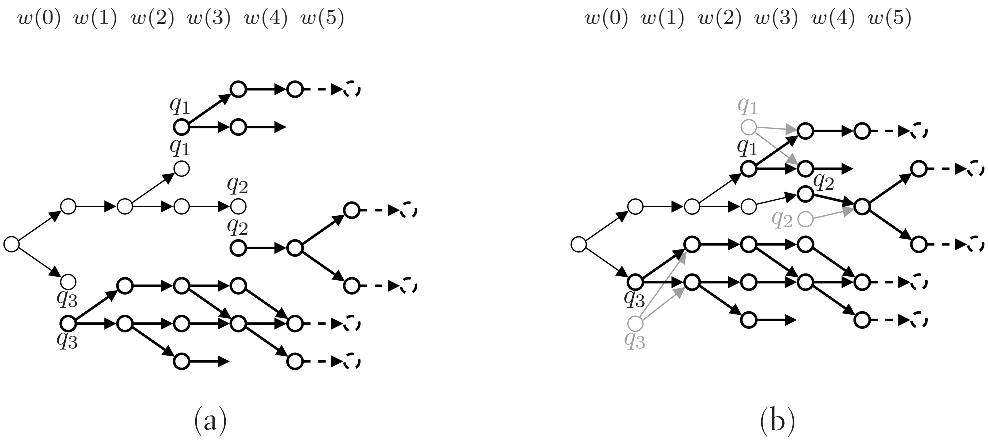


Fig. 2.5: Extending a partial run of an automaton on an input w into a run of the automaton using runs of the automaton's subautomata. (a) A partial run of an automaton on w together with runs of its subautomata on suffixes of w; (b) The run (black nodes and edges) obtained by joining the runs of the subautomata with the partial run of the automaton)

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

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

We claim that G' is an inf- (fin-)accepting run of $\mathcal{A}$ on w. First, V' can be partitioned into finite disjoint levels by defining $V_i' \stackrel{\text{def}}{=} V_i \cup \bigcup_{0 \leq j < i} \bigcup_{v \in \widehat{V}_j} V_{i-j}^v$; then also $E' \subseteq \bigcup_{i=0}^{\omega} (V_i' \times 2^{V_{i+1}'})$ holds.

Let $v \in V'$. If $v \in V$, and v has an outgoing edge in V, then the fact that G is a partial run (together with the definition of G') guarantees the existence of a unique edge $e \in E \subseteq E'$ such that this edge is labelled consistently with a transition of $\mathcal{A}$.

Otherwise, if $v \in V$ has no outgoing edges, then $v \in \widehat{V}_i$ for some $0 \leq i < \omega$. Because G^v is a run of $\mathcal{A}^{L(v)}$ on w^i, there exists a unique edge $e^v = \langle v_0^v, V_1^v \rangle \in E^v$ labelled (in G^v) with a transition $t = \big\langle L^v(v_0^v), \Gamma, F, L^v(V_1^v) \big\rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w^i(0) = w(i) \in \Gamma$. Thus, by the definition of G', $V_1^v \subseteq V'$ (because $v_0^v \notin V_1^v$), and E' contains the unique edge $e = \langle v, V_1^v \rangle$. Furthermore, because $L'(v) = L(v) = L^v(v_0^v)$ and $L'(v') = L^v(v')$ for all $v' \in V_1^v$, $t = \big\langle L^v(v_0^v), \Gamma, F, L^v(V_1^v) \big\rangle =$

$\langle L'(v), \Gamma, F, L'(V_1^v)\rangle$, and because $L'(e) = L^v\big(\langle v_0^v, V_1^v\rangle\big) = L^v(e^v) = t$, the labelling of e is consistent in G'.

Finally, if $v \in V^{v'} \setminus \{v_0^{v'}\}$ for some $0 \leq i < \omega$ and $v' \in \widehat{V}_i$, then, because $G^{v'}$ is a run of $\mathcal{A}^{L(v')}$ on w^i, there exists a unique edge $e^v = \langle v, \widetilde{V}^v\rangle \in E^v$ labelled (in G^v) with a transition $t = \big\langle L^v(v), \Gamma, F, L^v(\widetilde{V}^v)\big\rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w^i(0) = w(i) \in \Gamma$. Because $v_0^{v'} \notin \widetilde{V}^v$ and $v \neq v_0^{v'}$, $\widetilde{V}^v \subseteq V'$ and $e^v \in E'$ hold by the definition of G', and e^v is still unique in G'. Since $L'(v') = L^v(v')$ holds for all $v' \in \{v\} \cup \widetilde{V}^v$, it follows that $t = \big\langle L^v(v), \Gamma, F, L^v(\widetilde{V}^v)\big\rangle = \big\langle L'(v), \Gamma, F, L^v(\widetilde{V}^v)\big\rangle$, and since $L'(e^v) = L^v(e^v) = t$, the labelling L' is consistent.

Let $v' \in V' \setminus V_0$. If $v' \in V$, then, because G is a partial run, v' is a successor of some node $v \in V \subseteq V'$ in G, and because $E \subseteq E'$, the same still holds in G'. If $v' \in V^v \setminus \{v_0^v\}$ for some $v \in \bigcup_{0 \leq i < \omega} \widehat{V}_i$, then $v' \in V_j^v$ for some $1 \leq j < \omega$ by the definition of G'. If $j > 1$, then v' is a successor of some node $v'' \in V_{j-1}^v \subseteq V'$ in G^v, and because $\bigcup_{1 \leq i < \omega}(V_i^v \times 2^{V_{i+1}^v}) \subseteq E'$, the same holds also in G'. Otherwise, v' is a successor of a node $v'' \in \bigcup_{0 \leq i < \omega} \widehat{V}_i$ by the definition of G', and the third condition of a run is satisfied also in this case. It follows that G' is a run of $\mathcal{A}$ on w.

Let $r \in \mathcal{E}(G')$ be an acceptance sequence of G'. If r is contained in G, then r satisfies the inf- (fin-)acceptance condition by assumption. Otherwise the sequence consists of finitely many (possibly none) consecutive edges in G followed by an edge of the form $e = \langle v, V_1^v\rangle$ for some $v \in \bigcup_{0 \leq i < \omega} \widehat{V}_i$ and $V_1^v \subseteq V^v$, which is then followed by an infinite sequence of consecutive edges through G^v forming an infinite suffix of some acceptance sequence r^v of G^v. Since the number of edges in r preceding this suffix is finite, it follows that $\inf(r) = \inf(r^v)$ and $\operatorname{fin}(r) = \operatorname{fin}(r^v)$, and since either $\inf(r^v) = \mathcal{F}$ or $\operatorname{fin}(r^v) = \emptyset$ (depending on the acceptance mode), it follows that r satisfies the same acceptance condition. We conclude that G' is an accepting run of $\mathcal{A}$ on w. □

Our last result in this section shows that the language inf- or fin-accepted by an alternating automaton depends only on the subautomaton rooted at the initial state of the automaton. In other words, 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 start state or in their target states) without changing the language of the automaton.

Proposition 2.3.11 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ be an alternating automaton. Then, for all $w \in \Sigma^\omega$, $\mathcal{A}$ inf- (fin)-accepts w iff $\mathcal{A}^{q_I}$ inf- (fin-)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 the same set of acceptance conditions, every inf- (fin-)accepting run of $\mathcal{A}^{q_I}$ on some $w \in \Sigma^\omega$ is also an inf- (fin-)accepting run of $\mathcal{A}$ on w.

Conversely, if $G = \langle V, E, L\rangle$ (with $V_0 = \{v_0\}$) is an inf- (fin-)accepting run of $\mathcal{A}$ on $w \in \Sigma^\omega$, then, because each node $v \in V$ is either v_0 or a descendant of v_0 in G, $L(v)$ is either q_I or a descendant of q_I in $\mathcal{A}$ by Proposition 2.3.6. This implies also that no edge $e \in E$ can be labelled with a transition that has a state that is not reachable from q_I as its start state or in its target states. But then, by the definition of $\mathcal{A}^{q_I}$, all labels of the nodes and edges in G are (consistently labelled) states and transitions of $\mathcal{A}^{q_I}$, and thus G is an inf- (fin-)accepting run of $\mathcal{A}^{q_I}$ on w. □

Example 2.3.12 Consider again the alternating automaton depicted in Fig. 2.1 (a) (page 14). If we choose q_4 instead of q_1 as the initial state of this automaton, then, by Proposition 2.3.11, we know that the language of the automaton is completely determined by the subautomaton $\mathcal{A}^{q_4}$ shown in Fig. 2.1 (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) accept the same language. ■

2.3.2 Nondeterministic Automata

We say that the alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ is *nondeterministic* if $|Q'| = 1$ holds for all transitions $\langle q, \Gamma, F, Q' \rangle \in \Delta$. Therefore, if $G = \langle V, E, L \rangle$ is a run of a nondeterministic automaton, each level of G consists of identically labelled nodes.

2.3.3 Linear Alternating Automata

In this work we concentrate on a restricted set of alternating automata known as the class of *linear* or *very weak* alternating automata. These automata are known to be closely related to linear temporal logic in that every language definable as the set of models of an LTL formula is also a language recognized by some linear alternating automaton and vice versa [47, 58]. Since automata in general possess intuitively appealing "operational" characteristics, translating linear 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 may allow us to make the checking procedure more efficient.

Linear alternating automata have previously appeared in the works of Isli [37] and Rohde [58], both of whom referred to them as *very weak* alternating automata (a subclass of *weak* alternating automata introduced by Muller, Saoudi and Schupp [51]), and Löding and Thomas [47], who called these automata *linear* alternating automata. We shall use the terminology of Löding and Thomas in further discussion. Because our basic definitions of automata and acceptance generalize traditional definitions by allowing the automata to have multiple acceptance conditions associated with their transitions, we shall rephrase several basic results on linear 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.

Formally, a *linear alternating automaton* $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ is an alternating automaton for which there exists a mapping $\rho : Q \to \mathbb{N}$ such that for all transitions $\langle q, \Gamma, F, Q' \rangle \in \Delta$ for some $q \in Q$, $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$, $\rho(q') < \rho(q)$ holds for all $q' \in Q' \setminus \{q\}$.

The existence of the function ρ corresponds to the following characteristic structural property of linear alternating automata. (This result also establishes the equivalence of our definition with more traditional definitions obtained as a special case of weak alternating automata [51].)

Proposition 2.3.13 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton. Then, $\mathcal{A}$ is a linear alternating automaton iff all cycles in $\mathcal{A}$ are self-loops.*

Proof: ("$\Rightarrow$") Assume that $\mathcal{A}$ is a linear alternating automaton. Let $\rho : Q \to \mathbb{N}$ be a mapping satisfying the above criterion. Let $x = (q_i)_{0 \leq i \leq n}$ $(1 \leq n < \omega)$ be a cycle in $\mathcal{A}$. Assume that the cycle is not a self-loop, i.e., $n > 1$. Since q_{i+1} is a successor of q_i for all $0 \leq i < n$ 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.

("$\Leftarrow$") Assume that all 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 $q \in Q_0$, 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 $(q_i)_{0 \leq i < \omega}$ with $q_0 = q$ and $q_i \neq q_{i+1}$ for all $0 \leq i < \omega$. Since Q is finite, there would now exist two indices $0 \leq n < \omega$ and $n + 1 < m < \omega$ such that $q_{n+1} \neq q_n$ and $q_m = q_n$. However, the automaton would then contain a cycle $(q_i)_{n \leq i \leq m}$ that is not a self-loop, which is a contradiction.

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

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

is well-defined on Q.

To show that this function satisfies the criterion required of a mapping associated with a linear alternating automaton, assume that $q \in Q$ and $\langle q, \Gamma, F, Q' \rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$. Assume also that there exists a state $q' \in Q' \setminus \{q\}$ with $\rho(q') \geq \rho(q)$. 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' is a successor of q, there exists 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, which is a contradiction. Thus, $\rho(q') < \rho(q)$. □

Example 2.3.14 Figure 2.6 shows a linear alternating automaton without transition labels. The structure defined by the states and transitions of a linear alternating automaton can be considered to have been built from a directed acyclic graph by adding to it edges including their own start state in their target states. ■

Because all loops of linear alternating automata visit a single state of the automaton, every infinite path through a run of a linear alternating automaton will converge to a fixed state of the automaton after visiting some finite number of other states of the automaton.

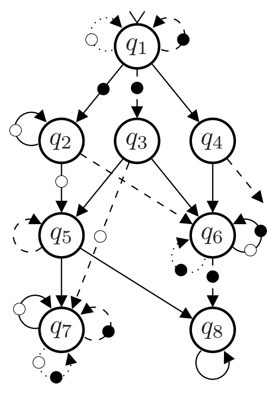


Fig. 2.6: A linear alternating automaton

Proposition 2.3.15 *Let $G = \langle V, E, L\rangle$ be a run of a linear alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$. For each acceptance sequence $(e_i)_{0\le i<\omega} = \big(\langle v_i, V_i'\rangle\big)_{0\le i<\omega} \in \mathcal{E}(G)$, there exists an index $0 \le j < \omega$ and a state $q \in Q$ such that for all $j \le k < \omega$, $L(v_k) = q$, and $L(e_k)$ is a self-loop transition of $\mathcal{A}$ with start state q.*

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

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

A corollary of Proposition 2.3.15 is that every infinite path of a fin-accepting run of a linear alternating automaton will converge to a nontransient state of the automaton.

Corollary 2.3.16 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ be a linear 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 acceptance sequence $r = (e_i)_{0\le i<\omega} = \big(\langle v_i, V_i'\rangle\big)_{0\le i<\omega} \in \mathcal{E}(G)$ converges to a nontransient state of $\mathcal{A}$.*

Proof: By Proposition 2.3.15, there exists a state $q \in Q$ and an index $0 \le j < \omega$ such that for all $j \le k < \omega$, $L(v_k) = q$, and $L(e_k)$ is a self-loop of $\mathcal{A}$ with start state q. Because G is fin-accepting, $\mathrm{fin}(r) = \emptyset$, and thus, for all $f \in \mathcal{F}$ and $j \le k < \omega$, there exists a $k < k' < \omega$ such that the self-loop $L(e_{k'}) \in \Delta$

does not include f in its acceptance conditions. 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 2.3.15 is that no acceptance condition associated with a non-self-loop transition of a linear alternating automaton affects the language recognized by the automaton. Thus, we can always remove all acceptance conditions from the non-self-loops of the automaton.

Corollary 2.3.17 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton. Define the 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 (i.e., $\Delta' \overset{def}{=} \big\{ \langle q, \Gamma, F, Q' \rangle \in \Delta \,\big|\, q \in Q, \Gamma \subseteq \Sigma, F \subseteq \mathcal{F}, Q' \subseteq Q, q \in Q' \big\} \cup \big\{ \langle q, \Gamma, \emptyset, Q' \rangle \,\big|\, \langle q, \Gamma, F, Q' \rangle \in \Delta$ for some $q \in Q, \Gamma \subseteq \Sigma, F \subseteq \mathcal{F}, Q' \subseteq Q \setminus \{q\} \big\}$). Then, for all $w \in \Sigma^\omega$, $\mathcal{A}'$ inf- (fin-)accepts w iff $\mathcal{A}$ inf- (fin-)accepts w.*

Proof: Let $r = (e_i)_{0 \le i < \omega} \in \mathcal{E}(G)$ be an acceptance sequence through a run G of either of the automata. By Proposition 2.3.15, there exists an index $0 \le j < \omega$ such that $L(e_i)$ is a self-loop of both automata for all $j \le i < \omega$, and thus r contains only finitely many non-self-loops. Therefore, none of these transitions can contribute to the acceptance conditions occurring infinitely often in the labels of the edges of r, i.e., $\inf(r) = \inf\big((e_i)_{j \le i < \omega}\big)$ and $\mathrm{fin}(r) = \mathrm{fin}\big((e_i)_{j \le i < \omega}\big)$. 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 2.3.18 Consider again the linear alternating automaton in Fig. 2.6. By Corollary 2.3.17, we can remove all acceptance conditions from the non-self-loop transitions of the automaton. This simplification results in the automaton shown in Fig. 2.7. 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 2.3.16, every acceptance sequence in a fin-accepting run of either automaton has to converge to one of the states q_1, q_5, q_7 or q_8. ■

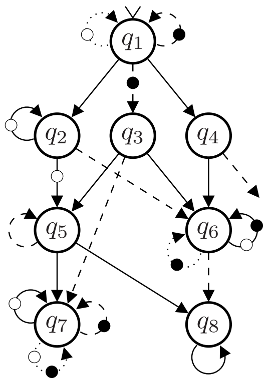


Fig. 2.7: The linear alternating automaton of Fig. 2.6 after removing acceptance conditions from its non-self-loop transitions

3 BASIC AUTOMATON TRANSLATION

The languages definable by linear temporal logic formulas are known to be recognizable by nondeterministic automata on infinite words [75, 76], i.e., every linear 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 into automata [14, 15, 24, 28, 29, 30, 31, 32, 37, 43, 59, 60, 62, 68, 75, 78], along with techniques for minimization of the automata [20, 21, 22, 25, 34, 62].

A direct translation of an LTL formula φ into a nondeterministic automaton may result in an automaton, the size of which is exponential in the number of subformulas in the formula φ [19, 75, 76]. However, the addition of universal choice to the automata allows translations into alternating automata with only a linear number of states in the size of the input formula [24, 28, 37, 48, 58, 72]. While all of these translations are very similar (basically, a linear translation corresponds to a rewriting procedure for LTL formulas), 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 [37, 72] (which essentially consists of the subformulas of the formula and their negations), rewriting the formula in positive normal form before translation [24, 28, 48], or using a complementation procedure for alternating automata [58].

In this chapter we describe a translation from linear temporal logic to linear alternating automata working in fin-acceptance mode. Borrowing ideas from known translation procedures [28, 58], we give a set of rules for translating the positive normal form of a formula $\varphi \in LTL(AP)$ into an equivalent linear alternating automaton $\mathcal{A}_\varphi$ over the fixed alphabet 2^{AP} in a bottom-up manner by joining automata built recursively for subformulas of φ into increasingly complex automata. 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 [28]) gives a simple explanation for the introduction of new acceptance conditions during the translation. We show that the worst-case size of the resulting automaton meets the best upper bound known for similar translations presented in the literature and show the correctness of the translation.

3.1 REPRESENTING TRANSITION GUARDS

We first review the notation that is customarily used [21, 28] to simplify the representation of transition guards of automata 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* ψ, for example, $\psi \stackrel{def}{=} \bigvee_{i=1}^{n} \big((\bigwedge_{p \in \sigma_i} p) \land (\bigwedge_{p \in AP \setminus \sigma_i} \neg p)\big)$, such that, given $\sigma \subseteq AP$, $\sigma \models \psi$ iff $\sigma \in \Gamma$; conversely, each Boolean formula θ is characteristic for the family of its models $\Gamma_\theta \stackrel{def}{=} \{\sigma \subseteq AP \mid \sigma \models \theta\} \in 2^{2^{AP}}$. Therefore, when considering the runs of an alternating automaton, the requirement 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}.

3.2 TRANSLATION RULES

Let φ be an LTL formula in positive normal form. 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 $\mathcal{A}_\varphi = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$, where $Q \stackrel{def}{=} \{q_I\}$, $\Delta \stackrel{def}{=} \big\{\langle q_I, \varphi, \emptyset, \emptyset \rangle\big\}$, and $\mathcal{F} \stackrel{def}{=} \emptyset$.

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

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

Binary Connectives
Let $\varphi = (\varphi_1 \circ \varphi_2)$ for some binary connective $\circ \in \{\lor, \land, \mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}$. Let $\mathcal{A}_{\varphi_1} = \langle \Sigma, Q_1, \Delta_1, q_{I1}, \mathcal{F}_1 \rangle$ and $\mathcal{A}_{\varphi_2} = \langle \Sigma, Q_2, \Delta_2, q_{I2}, \mathcal{F}_2 \rangle$ be already

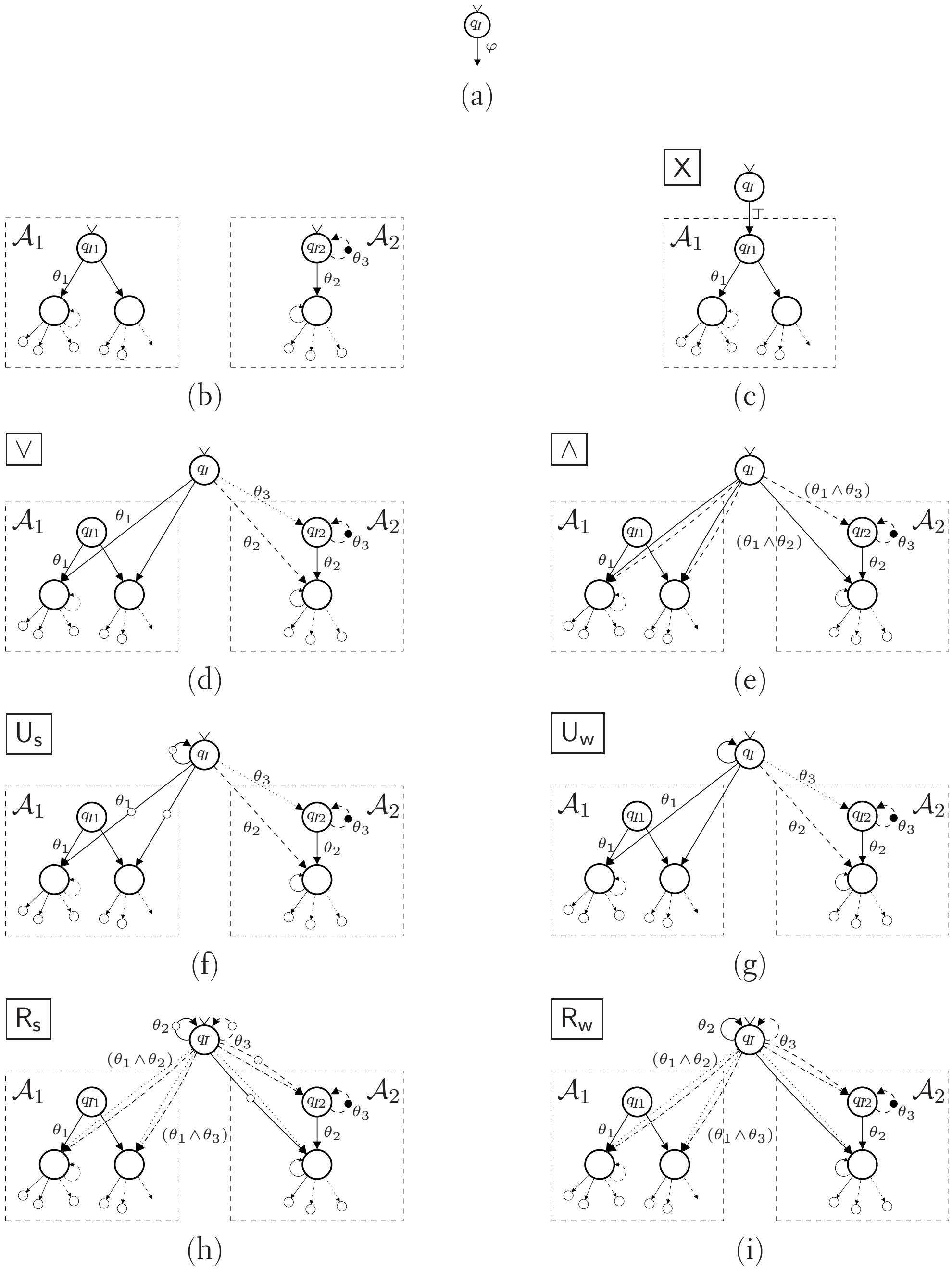


Fig. 3.1: Automata built using translation rules. (a) Automaton built from an atomic formula φ; (b) Two component automata $\mathcal{A}_1$ and $\mathcal{A}_2$; (c) Automaton built from $\mathcal{A}_1$ with the Next Time rule; (d)–(i) Automata built from $\mathcal{A}_1$ and $\mathcal{A}_2$ using the translation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

defined for the top-level subformulas φ_1 and φ_2 of φ, respectively, such that $\mathcal{A}^q_{\varphi_1} = \mathcal{A}^q_{\varphi_2}$ holds for all $q \in Q_1 \cap Q_2$ (i.e., if the two automata share a state, then the automata also share all states and transitions reachable from this state). The automaton $\mathcal{A}_\varphi = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ 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 $Q_1 \cup Q_2$;
- $\mathcal{F} \stackrel{def}{=} \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_\circ$; and
- $\Delta \stackrel{def}{=} \Delta_1 \cup \Delta_2 \cup \Delta_\circ$,

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

Simple Observations

Correspondence Between Subformulas of φ and States of $\mathcal{A}_\varphi$ The construction of an automaton $\mathcal{A}_\varphi$ for the given LTL formula φ 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 that do not share any subformulas can be constructed in either order), automata built for two identical subformulas of φ are nevertheless easily seen to be isomorphic. 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. It follows that the number of rule applications required equals the number of (syntactically) different subformulas of φ. This immediately proves the eventual termination of the formula translation due to the finiteness of $\mathrm{Sub}(\varphi)$, and, because each step of the translation adds exactly one new state to the result, there is a bijective correspondence between $\mathrm{Sub}(\varphi)$ and the state set of the final automaton.

Interpretation of the Translation Rules The correspondence between $\mathrm{Sub}(\varphi)$ and the states of the automaton $\mathcal{A}_\varphi$ gives a simple interpretation of each translation rule. Intuitively, the translation rules describe how to join the automata built for the top-level subformulas of a given LTL formula φ into an automaton that encodes in its structure "instructions" on how to run its components to recognize the language $\mathcal{L}(\varphi)$ (the proof that this is indeed the case follows in Sect. 3.4). Thus, for example, the translation rule for constructing an automaton $\mathcal{A}_{(\varphi_1 \wedge \varphi_2)}$ for the language $\mathcal{L}\big((\varphi_1 \wedge \varphi_2)\big)$ 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 effectively 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 $\mathcal{A}_{(\varphi_1 \wedge \varphi_2)}$ mimic a pair of initial transitions taken synchronously by each of the component automata. As a result, the initial transition in any run of $\mathcal{A}_{(\varphi_1 \wedge \varphi_2)}$ corresponds to spawning both $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ on the same input. 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 state of the automaton after the transition matches the collective state

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)

$\circ$	$\mathcal{F}_\circ$	$\Delta_\circ$
$\vee$	$\emptyset$	$\Big\{\langle q_I, \theta_1, \emptyset, Q'_1\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\Big\} \cup \Big\{\langle q_I, \theta_2, \emptyset, Q'_2\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\Big\}$
$\wedge$	$\emptyset$	$\left\{\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q'_1 \cup Q'_2\rangle \;\middle\vert\; \begin{array}{l}\langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\end{array}\right\}$
$\mathsf{U_s}$	$\{f\}$	$\Big\{\langle q_I, \theta_1, \{f\}, Q'_1 \cup \{q_I\}\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\Big\} \cup \Big\{\langle q_I, \theta_2, \emptyset, Q'_2\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\Big\}$
$\mathsf{U_w}$	$\emptyset$	$\Big\{\langle q_I, \theta_1, \emptyset, Q'_1 \cup \{q_I\}\rangle \;\Big\vert\; \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\Big\} \cup \Big\{\langle q_I, \theta_2, \emptyset, Q'_2\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\Big\}$
$\mathsf{R_s}$	$\{f\}$	$\Big\{\langle q_I, \theta_2, \{f\}, Q'_2 \cup \{q_I\}\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\Big\} \cup \left\{\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q'_1 \cup Q'_2\rangle \;\middle\vert\; \begin{array}{l}\langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\end{array}\right\}$
$\mathsf{R_w}$	$\emptyset$	$\Big\{\langle q_I, \theta_2, \emptyset, Q'_2 \cup \{q_I\}\rangle \;\Big\vert\; \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\Big\} \cup \left\{\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q'_1 \cup Q'_2\rangle \;\middle\vert\; \begin{array}{l}\langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\end{array}\right\}$

reached by $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ 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 needs to be applied to the initial transitions of an automaton constructed 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 effectively 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, to construct an automaton for the formula $(\varphi_1 \,\mathsf{U_s}\, \varphi_2)$, first build automata for the top-level subformulas and then join them into an automaton that verifies that either φ_2 holds for the infinite suffix of the input beginning at the current input position, or that φ_1 holds for this suffix and $(\varphi_1 \,\mathsf{U_s}\, \varphi_2)$ still holds for 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 $\mathcal{L}(\varphi_1)$, while 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 results in 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 discarded by Proposition 2.3.11 without any change in the language of the automaton. This simplification then gives the rules shown in Table 3.1.

Linearity 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 none of the rules manipulate the transition relation of any component automaton, this implies—by induction—that every state q of the automaton $\mathcal{A}_\varphi = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ 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 arise in the application of the translation rules to subformulas of φ with a binary temporal main connective. By these observations and Proposition 2.3.13, it follows that the automaton constructed by the translation rules is a linear alternating automaton.

Structure of Transition Guards The guard of the only transition in an automaton built for any atomic formula is simply the formula itself, encoding the subsets of AP that satisfy it: for $\top$, all subsets of AP; for $\bot$, no subsets of AP; for literals, all subsets of AP that (do not) include the (negated) proposition forming 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. By the translation rules, 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 AP by the semantics of $\wedge$. This very restricted form allows for efficient checking of propositional implications between the guards, which is needed when simplifying the automaton. This will be discussed in the next chapter.

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. Thus, the number of acceptance conditions in the final automaton equals the number of syntactically distinct strong eventuality subformulas of the given formula. Intuitively, since 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. 3.4.

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 2.3.17. 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 start state.[1]

[1]Actually, this fact can be used (together with Proposition 2.3.15) 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 [28]. We shall not do this here, however, since the correctness of some of the simplification heuristics to be presented in the next chapter relies on the strict correspondence between the acceptance conditions and the different temporal eventualities.

3.3 SIZE OF THE AUTOMATON $\mathcal{A}_\varphi^{q_I}$

As noted above, the automaton translation for an LTL formula φ requires $|\mathrm{Sub}(\varphi)|$ applications of a translation rule. Since each rule application adds one new state to the result, the translation ends with an automaton having exactly $|\mathrm{Sub}(\varphi)|$ states, which also gives a simple upper bound for the size of an automaton corresponding to the formula φ.

By Proposition 2.3.11, the language of an alternating automaton does not depend on those non-initial states of the automaton that are not reachable from the initial state of the automaton. However, this fact is not taken into account in the above upper bound given for the size of an automaton $\mathcal{A}_\varphi$ built for a given LTL formula, since the bound is only indirectly obtained from the number of steps required for the translation. By Proposition 2.3.11, a tighter bound can be given by considering the size of the subautomaton $\mathcal{A}_\varphi^{q_I}$ obtained from $\mathcal{A}_\varphi$ by restricting $\mathcal{A}_\varphi$ to the set of states including 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 the transitions leaving the initial state 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 changes—or even refers 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 reachable from 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 occurs (by the definition of the translation rules) iff q_I is the initial state of an automaton corresponding to 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 $\mathsf{X}\varphi_1 \in \mathrm{Sub}(\varphi)$. (Since $\mathsf{X}\varphi_1 \in \mathrm{Sub}(\varphi)$, the Next Time rule will eventually have to be applied to the automaton $\mathcal{A}_{\varphi_1}$; 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 reachable from the initial state of the final automaton. Assume that q_I

is the initial state of an automaton (corresponding to a formula $\varphi_1 \in \mathrm{Sub}(\varphi)$) such that q_I satisfies none of the above conditions. Then, φ has at least one subformula with φ_1 as a top-level subformula. Because $\mathsf{X}\varphi_1 \notin \mathrm{Sub}(\varphi)$, 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 $\mathsf{X}\varphi_1 \notin \mathrm{Sub}(\varphi)$, 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 3.3.1 *Let $\mathcal{A}_\varphi$ be the alternating automaton built for the LTL formula φ using the translation rules. Then,*

$$|\mathcal{A}_\varphi^{q_I}| = \left| \begin{array}{l} \{\varphi\} \\ \cup \big\{(\varphi_1 \circ \varphi_2) \in \mathrm{Sub}(\varphi) \;\big|\; \circ \in \{\mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}\big\} \\ \cup \big\{\varphi_1 \in \mathrm{Sub}(\varphi) \;\big|\; \mathsf{X}\varphi_1 \in \mathrm{Sub}(\varphi)\big\} \end{array} \right|$$

This result leads to the following upper bound for the size of 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 implicitly given by Gastin and Oddoux in their paper [28], and it is also closely related to a known upper bound for translating LTL directly into nondeterministic automata [14].

Corollary 3.3.2 *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 with at most $1 + |\mathrm{Temp}(\varphi)|$ states. (If φ is a binary pure temporal formula, the upper bound reduces to $|\mathrm{Temp}(\varphi)|$.)*

Proof: As noted in Sect. 2.2.3, the positive normal form φ' of φ has at most as many pure temporal subformulas as φ, i.e., $|\mathrm{Temp}(\varphi')| \leq |\mathrm{Temp}(\varphi)|$. By applying the translation to φ', the result follows directly from Proposition 3.3.1 by observing that

$$\left| \begin{array}{l} \big\{(\varphi_1 \circ \varphi_2) \in \mathrm{Sub}(\varphi') \;\big|\; \circ \in \{\mathsf{U_s}, \mathsf{U_w}, \mathsf{R_s}, \mathsf{R_w}\}\big\} \\ \cup \big\{\varphi_1 \in \mathrm{Sub}(\varphi') \;\big|\; \mathsf{X}\varphi_1 \in \mathrm{Sub}(\varphi')\big\} \end{array} \right| \leq |\mathrm{Temp}(\varphi')|.$$

□

3.4 CORRECTNESS OF THE TRANSLATION

In this section we show the correctness of the translation. We start by proving two lemmas needed in the correctness proof. The first of these establishes a basic correspondence between a single step of operation of an alternating automaton and inf- (fin-)acceptance: the automaton accepts its input only if its first transition spawns copies of the automaton, all of which accept the remainder of the input following the symbol consumed by taking the transition. Actually, this result has also a converse: the automaton can always be made to accept its input if it has such an initial transition synchronizable with the first symbol of the input.

Lemma 3.4.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton, and let $w \in \Sigma^\omega$. Then, $\mathcal{A}$ inf- (fin-)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 $w(0) \in \Gamma$ and for all $q \in Q'$, the subautomaton $\mathcal{A}^q$ inf- (fin-)accepts w^1.*

Proof: ("$\Rightarrow$") Assume that $\mathcal{A}$ inf- (fin-)accepts w. Then $\mathcal{A}$ has an inf- (fin-)accepting run $G = \langle V, E, L \rangle$ on w. Thus there exists a state $v_0 \in V_0$ labelled with the initial state of $\mathcal{A}$ and an edge $e = \langle v_0, V_1 \rangle \in E$ labelled 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 $Q' = L(V_1)$. 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 inf- (fin-)accepting, each of these runs is also inf- (fin-)accepting by Proposition 2.3.9.

("$\Leftarrow$") 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 $w(0) \in \Gamma$, and the subautomaton $\mathcal{A}^{q_i}$ has an inf- (fin-)accepting run on w^1 for all $1 \leq i \leq n$. Define the graph $G = \langle V, E, L \rangle$, where

- $V \overset{def}{=} \{v_I, v_1, \ldots, v_n\}$, where $v_I \neq v_i$ for all $1 \leq i \leq n$ (and $v_i \neq v_j$ for all $1 \leq i, j \leq n$, $i \neq j$),
- $E \overset{def}{=} \big\{ \langle v_I, \{v_1, \ldots, v_n\} \rangle \big\}$, and
- $L(v_I) \overset{def}{=} q_I$, $L(v_i) \overset{def}{=} q_i$ for $1 \leq i \leq n$, and $L\big(\langle v_I, \{v_1, \ldots, v_n\} \rangle\big) \overset{def}{=} t$.

It is easy to see from the definitions that G is a partial run of $\mathcal{A}$ on w: obviously, V consists of finite disjoint levels $V_0 = \{v_I\}$, $V_1 = \{v_1, \ldots, v_n\}$, $V_i = \emptyset$ for all $2 \leq i < \omega$, the labelling of the only edge in G is consistent, and each node $v \in V \setminus \{v_I\}$ is a successor of v_I. Since there exist no infinite sequences of consecutive edges through G, G is trivially inf- (fin-)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 an inf- (fin-)accepting run on w^1 for all $1 \leq i \leq n$ by the assumption, we can apply Proposition 2.3.10 to extend G into an inf- (fin-)accepting run of $\mathcal{A}$ on w. $\square$

The following lemma characterizes fin-acceptance in a linear 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 3.4.2 *Let $\varphi = (\varphi_1 \circ \varphi_2) \in LTL(AP)$ $(\circ \in \{\mathsf{U_s}, \mathsf{U_w}\})$, 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 linear alternating automata constructed using the translation* *rules for φ, φ_1 and φ_2, respectively. Then, 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-accept(s) 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: ("$\Rightarrow$") 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 sequence of consecutive 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 $i < \omega$ ($L(e_i)$ is not a self-loop), the transition $L(e_i)$ corresponds to some initial transition of $\mathcal{A}_2$. Let $0 \leq j < i + 1$, and let $L(e_j) = L\big(\langle v_j, V'_j\rangle\big) = t = \langle q_I, \theta, F, Q'\rangle \in \Delta$ for some $\theta \in PL(AP)$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$. Because G is a run, $w(j) \models \theta$ and $Q' = L(V'_j)$. We consider the above two cases separately.

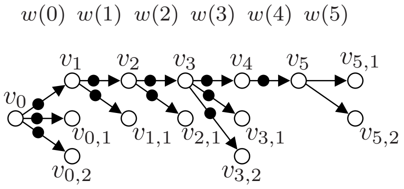
- If t is a self-loop of $\mathcal{A}$, there exists a transition $\langle q_{I1}, \theta, F_1, Q'_1\rangle \in \Delta_1$ for some $F_1 \subseteq \mathcal{F}_1$ and $Q'_1 \subseteq Q_1$ such that $Q' = Q'_1 \cup \{q_I\}$.

  Because G is a fin-accepting run of $\mathcal{A}$, 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 $Q'_1 \subseteq Q' = L(V'_j)$, it follows that $\mathcal{A}^{q'} = \mathcal{A}_1^{q'}$ has a fin-accepting run on w^{j+1} for all $q' \in Q'_1$. Moreover, because $w(j) = (w^j)(0) \models \theta$, Lemma 3.4.1 shows that $\mathcal{A}_1$ has a fin-accepting run on w^j.

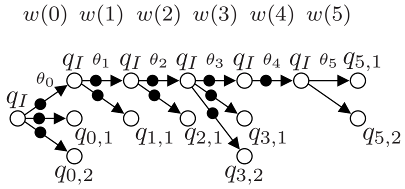
- If t is not a self-loop of $\mathcal{A}$, then t corresponds to an initial transition $\langle q_{I2}, \theta, F_2, Q'\rangle \in \Delta_2$ for some $F_2 \subseteq \mathcal{F}_2$, and thus $Q' \subseteq Q_2$. Similar to the self-loop case, the subautomaton $\mathcal{A}^{L(v')}$, which equals $\mathcal{A}_2^{L(v')}$, fin-accepts w^{j+1} for all $v' \in V'_j$ by Proposition 2.3.9. Because $Q' = L(V'_j)$ and $w(j) \models \theta$, Lemma 3.4.1 shows that $\mathcal{A}_2$ fin-accept(s) w^j.

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 $\circ = \mathsf{U_s}$. For if this were the case, then $r = (e_j)_{0\leq j<i}$ would be an infinite sequence of consecutive edges of G labelled with initial self-loops of $\mathcal{A}$, and thus r would be an acceptance sequence through G. However, because all initial self-loops of $\mathcal{A}$ share a common acceptance condition if $\circ = \mathsf{U_s}$, $\text{fin}(r)$ 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 $i < \omega$, and the result follows.

("$\Leftarrow$") Assume that there either exists an index $0 \leq i < \omega$ such that $\mathcal{A}_2$ fin-accept(s) w^i and for all $0 \leq j < i$, $\mathcal{A}_1$ fin-accepts w^j, or that $\circ = \mathsf{U_w}$, and



(a)



(b)

Fig. 3.2: Construction of a partial run in Lemma 3.4.2 ($\circ = \mathsf{U_s}$, $i = 5$). (a) Node and edge structure; (b) Labelling of nodes and edges

$\mathcal{A}_1$ fin-accepts w^i for all $0 \le i < \omega$. That is, assume that there exists an index $0 \le i \le \omega$ such that $\mathcal{A}_1$ fin-accepts w^j for all $0 \le j < i$, and if $i < \omega$, then $\mathcal{A}_2$ fin-accept(s) w^i.

By Lemma 3.4.1, $\mathcal{A}_1$ has an initial transition $t_{1,j} = \langle q_{I1}, \theta_{1,j}, F_{1,j}, Q_{1,j}\rangle \in \Delta_1$ for some $\theta_{1,j} \in PL(AP)$, $F_{1,j} \subseteq \mathcal{F}_1$ and $Q_{1,j} \subseteq Q_1$ for all $0 \le j < i$ such that $w(j) \models \theta_{1,j}$, and $\mathcal{A}_1^{q'}$ fin-accepts w^{j+1} for all $q' \in Q_{1,j}$. Additionally, if $i < \omega$, an analogous result holds for an initial transition $t_{2,i} = \langle q_{I2}, \theta_{2,i}, F_{2,i}, Q_{2,i}\rangle \in \Delta_2$ 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 $\theta_j = \theta_{1,j}$, $F_j = \{f\}$ for some new acceptance condition f ($\circ = \mathsf{U_s}$) or $F_j = \emptyset$ ($\circ = \mathsf{U_w}$), and $Q_j = Q_{1,j} \cup \{q_I\}$ for all $0 \le j < i$. Furthermore, if $i < \omega$, then there exists also a transition $t_i = \langle q_I, \theta_i, \emptyset, Q_i\rangle \in \Delta$, where $\theta_i = \theta_{2,i}$ and $Q_i = Q_{2,i}$. It is easy to see that $w(j) \models \theta_j$ and $\mathcal{A}^{q'}$ fin-accepts w^{j+1} for all $q' \in Q_j \setminus \{q_I\}$ and $0 \le j < i+1$.

Because Q_j is finite, we write $Q_j \setminus \{q_I\} = \{q_{j,1}, q_{j,2}, \ldots, q_{j,n_j}\} \subseteq Q$ for all $0 \le j < i+1$ ($0 \le n_j < \omega$, $q_{j,k} \ne q_{j,l}$ for all $1 \le k, l \le n_j$, $k \ne l$).

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

Figure 3.2 shows a possible structure for G with $\circ = \mathsf{U_s}$ and $i = 5$.

With these definitions, G is a partial run of $\mathcal{A}$ on w:

- $V_0 = \{v_0\}$, $L(v_0) = q_I$, and V is partitioned into finite disjoint levels (with edges only between successive levels) by construction.
- Let $v \in V_j$ for some $0 \le j < \omega$. Then v either has no outgoing edges, or $v = v_j$. By the definition of E, v_j has the unique outgoing edge $e = \langle v_j, V_{j+1}\rangle \in E$. Because $L(e) = t_j = \langle q_I, \theta_j, F_j, Q_j\rangle = \big\langle L(v_j), \theta_j, F_j, L(V_{j+1})\big\rangle$ and $w(j) \models \theta_j$, the edge labelling is consistent.
- Let $v \in V_j$ for some $1 \le j < \omega$. Then v is clearly a successor of the node $v_{j-1} \in V_{j-1}$, and the third condition of a partial run is satisfied.

It follows that G is a partial run of $\mathcal{A}$ on w. If $i < \omega$, then the edge set E is finite. Therefore $\mathcal{E}(G) = \emptyset$, and G is trivially fin-accepting. Otherwise G contains a unique infinite sequence of consecutive edges, all of which are labelled with initial self-loops of $\mathcal{A}$. Because $i = \omega$ can hold only if $\circ = \mathsf{U_w}$, the set of acceptance conditions of each initial self-loop of $\mathcal{A}$ is empty. It follows that the edge sequence satisfies the fin-acceptance condition also in this case, and thus G is a partial fin-accepting run of $\mathcal{A}$ on w.

Let $v \in V_j$ be a node at level j of G for some $0 \leq j < \omega$ with no outgoing edges in G. Then $j \geq 1$, and $v = v_{j-1,k}$ for some $1 \leq k \leq n_{j-1}$. Because $L(v_{j-1,k}) \in Q_{j-1} \setminus \{q_I\}$ and $\mathcal{A}^{q'}$ $(= \mathcal{A}_1^{q'}$ or $\mathcal{A}_2^{q'})$ has a fin-accepting run on w^j for all $q' \in Q_{j-1} \setminus \{q_I\}$, it follows that G can be extended into a fin-accepting run of $\mathcal{A}$ on w by Proposition 2.3.10, and thus $\mathcal{A}$ fin-accepts w. □

Using the above lemmas, we can give a simple inductive proof of the correctness of the translation.

Theorem 3.4.3 *Let φ be an LTL formula, and let $\mathcal{A}_\varphi$ be the automaton constructed from φ using the translation rules. Then, for all $w \in (2^{AP})^\omega$, $\mathcal{A}_\varphi$ fin-accepts w iff $w \models \varphi$.*

Proof: We proceed by induction on the size of the formula φ. If $|\varphi| = 1$, it follows directly from the definition of $\mathcal{A}_\varphi$ and Lemma 3.4.1 that $\mathcal{A}$ has a fin-accepting run on w iff $w(0) \models \varphi$, which is equivalent to $w \models \varphi$, because φ is a Boolean formula in this case.

Assume that the result holds for all LTL formulas of size less than or equal to some fixed $1 \leq k < \omega$, and let φ be a compound LTL formula of size $|\varphi| = k+1$. 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 $q \in Q'$, $\mathcal{A}_\varphi^q$ fin-accepts w^1 *(Lemma 3.4.1)*

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 $q \in Q'$, $\mathcal{A}_\varphi^q$ fin-accepts w^1 *(Lemma 3.4.1)*

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

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 $q \in Q_2'$, $\mathcal{A}_\varphi^q$ $(= \mathcal{A}_{\varphi_2}^q)$ 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 *(Lemma 3.4.1)*

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 *(Lemma 3.4.1)*

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 *(Lemma 3.4.1)*

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 3.4.2)*

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 3.4.2)* iff there exists an index $0 \leq i < \omega$ such that $w^i \models (\varphi_1 \land \varphi_2)$ 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 "$\land$" and the induction hypothesis)

iff $w \models \big(\varphi_2 \circ (\varphi_1 \land \varphi_2)\big)$ *(semantics of LTL)*

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

The result holds by induction for all formulas $\varphi \in LTL(AP)$. □

3.5 REVERSE TRANSLATION

In this section we verify the result that for any linear alternating automaton $\mathcal{A}$ over an alphabet Σ with 2^n elements for some $n \in \mathbb{N}$ (i.e., a set that is isomorphic 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 $w \models \varphi$. This justifies the use of the term "linear" for our automata and shows that our linear alternating automata with multiple acceptance conditions on transitions are equally expressive to the variants previously presented in the literature. Proofs for this same result (based on slightly different basic definitions and varying interpretation of acceptance conditions) have previously been presented by Rohde [58], and Löding and Thomas [47]; in this section, we give a direct proof of this result for automata having multiple acceptance conditions on transitions instead of states.

As shown by Rohde [58], this result can be generalized to linear alternating automata over an arbitrary (finite) nonempty alphabet Σ in the sense that for any linear 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 $\big(\alpha(w(i))\big)_{0 \leq i < \omega} \models \varphi$. The existence of α is easily proved 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 result to the linear 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 with a result that characterizes (in LTL) the behavior of a linear alternating automaton $\mathcal{A} = \langle 2^S, Q, \Delta, q_I, \mathcal{F} \rangle$ looping through a state $q \in Q$ along a path of a fin-accepting run on some input $w \in (2^S)^\omega$. By interpreting this path as the stepwise behavior of a single copy of the automaton, we see that the copy either stays in the state q for a finite number of steps, exits the state, and never enters it again (because all loops in the automaton are self-loops), or it remains in the state indefinitely by taking only self-loop transitions. In this case, the acceptance sequence determined by these self-loops must not violate the fin-acceptance condition, because the run is fin-accepting. Hence, for all acceptance conditions $f \in \mathcal{F}$, the copy of the automaton needs to take infinitely many self-loops that do not include f in their acceptance conditions.

To formalize this intuition, we first introduce some notation. Assume that q is the start state of the i^{th} consecutive edge (labelled with a transition $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$) in a path through the fin-accepting run of $\mathcal{A}$. Because

q is the initial state of some subautomaton of $\mathcal{A}$, it follows by Lemma 3.4.1 that the guard Γ includes the input symbol $w(i) \in 2^S$ (i.e., $w(i) \models \theta$ holds for the characteristic Boolean formula 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., $w^{i+1} \models \varphi_{q'}$. We thus find that

$$w^i \models \mu(\theta, Q') \overset{def}{\equiv} \big(\theta \wedge \mathsf{X}(\bigwedge_{q' \in Q' \setminus \{q\}} \varphi_{q'})\big).$$

Using μ, the implications of $\mathcal{A}$ taking either a self-loop, a non-self-loop, or a self-loop not including f in its acceptance conditions can now be written as the LTL formulas

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

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

With these definitions, we can now give an LTL characterization of the above description of the looping behavior of $\mathcal{A}$. 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 3.5.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F}\rangle$ be a linear 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 $w \models \varphi_{q'}$. Then, for all $w \in \Sigma^\omega$, $\mathcal{A}^q$ fin-accepts w iff w satisfies the formula*

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

Proof: ("$\Rightarrow$") 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 $0 \le i \le \omega$ and a sequence $(e_j)_{0 \le j < i+1}$, $e_j = \langle v_j, V_j'\rangle \in E \cap (V_j \times 2^{V_{j+1}})$, of consecutive edges such that $L(e_j)$ is an initial self-loop of $\mathcal{A}^q$ for all $0 \le j < i$, 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 $0 \le j < i + 1$.

Let $0 \le j < i + 1$. Because G is a run, $e_j = \langle v_j, V_j'\rangle$ is labelled with a transition $\langle q, \theta_j, F_j, Q_j'\rangle \in \Delta$ such that $w(j) \models \theta_j$ and $Q_j' = L(V_j')$. Because θ_j is a Boolean formula, it follows that $w^j \models \theta_j$, 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 $w^{j+1} \models \varphi_{q'}$ for all $q' \in Q_j' \setminus \{q\}$. Therefore $w^j \models \big(\theta_j \wedge \mathsf{X}(\bigwedge_{q' \in Q_j' \setminus \{q\}} \varphi_{q'})\big)$, i.e., $w^j \models \mu(\theta_j, Q_j')$ 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 $\psi_{\text{self-loop}}$ that $w^j \models \psi_{\text{self-loop}}$ 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 $w^i \models \psi_{\text{non-self-loop}}$ holds. But then $w \models (\psi_{\text{self-loop}} \,\mathsf{U_s}\, \psi_{\text{non-self-loop}})$ holds by the semantics of LTL, which implies that $w \models \varphi_q$.

If $i = \omega$, then $(e_j)_{0 \leq j < i} = r$ is an acceptance sequence, all edges of which are labelled with initial self-loops of $\mathcal{A}^q$. As seen above, $w^j \models \mu(\theta_j, Q'_j)$ and $w^j \models \psi_{\text{self-loop}}$ hold for all $0 \leq j < \omega$. Fix $0 \leq j < \omega$, and let $f \in \mathcal{F}$. Because G is fin-accepting, $\text{fin}(r) = \emptyset$, and thus there exists an index $j < k < \omega$ such that $L(e_k)$ does not include f in its acceptance conditions. Because $w^k \models \mu(\theta_k, Q'_k)$ and $L(e_k)$ is an initial self-loop of $\mathcal{A}^q$, it follows that $w^k \models \psi_{\text{avoid}}(f)$. But then, because $k > j$, $w^j \models \mathsf{F}\psi_{\text{avoid}}(f)$, and because f is arbitrary and $w^j \models \psi_{\text{self-loop}}$, it follows that $w^j \models \big(\psi_{\text{self-loop}} \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(f)\big)$. Since j is arbitrary, we conclude that $w \models \varphi_q$ holds also in this case.

("$\Leftarrow$") Assume that $w \models \varphi_q$. Then w satisfies at least one of the formulas $(\psi_{\text{self-loop}} \,\mathsf{U_s}\, \psi_{\text{non-self-loop}})$ and $\mathsf{G}\big(\psi_{\text{self-loop}} \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(f)\big)$, and there necessarily exists a minimal index $0 \leq i \leq \omega$ such that $w^j \models \psi_{\text{self-loop}}$ holds for all $0 \leq j < i$, and if $i < \omega$, then $w^i \models \psi_{\text{non-self-loop}}$. From the definition of $\psi_{\text{self-loop}}$ it follows that the set of initial self-loop transitions $T_j \overset{def}{=} \big\{\langle q, \theta, F, Q' \rangle \in \Delta \,\big|\, q \in Q', w^j \models \mu(\theta, Q')\big\}$ is nonempty for all $0 \leq j < i$. Our goal is to choose self-loops $t_j \overset{def}{=} \langle q, \theta_j, F_j, Q_j \rangle \in T_j$ for all $0 \leq j < i$ (and if $i < \omega$, an additional non-self-loop transition $t_i \overset{def}{=} \langle q, \theta_i, F_i, Q_i \rangle \in \Delta$ such that $w^i \models \mu(\theta_i, Q_i)$; t_i exists because $w^i \models \psi_{\text{non-self-loop}}$ holds in this case) such that we can construct a partial fin-accepting run of $\mathcal{A}^q$ on w by forming a (possibly infinite) sequence of consecutive edges labelled with these transitions. For this purpose, we also fix an arbitrary 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 $\prec$.

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 $\alpha_k \overset{def}{=} \min\big(\{k-1\} \cup \big\{0 \leq l < k \,\big|\, \bigcap_{l \leq j \leq k-1} F_j \neq \emptyset\big\}\big)$ 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 $\alpha_k = k-1$ otherwise). Let $\widetilde{F}_k \overset{def}{=} \bigcap_{\alpha_k \leq j \leq k-1} F_j$ be the set of all acceptance conditions shared by these transitions, and define $\tilde{f}_k \overset{def}{=} \min_{\prec} \widetilde{F}_k$ for every nonempty $\widetilde{F}_k$. Now, let $t_k \overset{def}{=} \langle q, \theta_k, F_k, Q_k \rangle$ 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.

Without loss of generality, we may write $Q_j \setminus \{q\}$ as a 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 $0 \leq j < i+1$.

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 $i+1 < j < \omega$;
- $E \overset{def}{=} \bigcup_{0 \leq j < i+1} \{\langle v_j, V_{j+1} \rangle\}$;
- For all $0 \leq j < i+1$, let $L(v_j) \overset{def}{=} q$, $L(v_{j,k}) \overset{def}{=} q_{j,k}$ for all $1 \leq k \leq n_j$,

and $L(\langle v_j, V_{j+1}\rangle) \overset{def}{=} t_j$.

(The definition of G is completely analogous to Lemma 3.4.2; see also Fig. 3.2.)

With these definitions, G is a partial run of $\mathcal{A}^q$ on w:

- $V_0 = \{v_0\}$, $L(v_0) = q$, V is partitioned into finite disjoint levels, and E consists of edges between successive levels of G.

- Let $v \in V_j$ for some $0 \leq j < \omega$. Then v either has no outgoing edges, or $v = v_j$ and $j < i + 1$. In the latter case, $L(v) = q$, and v has the unique outgoing edge $e = \langle v_j, V_{j+1}\rangle \in E$. Since $w^j \models \mu(\theta_j, Q_j)$, i.e., $w^j \models \big(\theta_j \wedge \mathsf{X}(\bigwedge_{q' \in Q_j \setminus \{q\}} \varphi_{q'})\big)$, $w(j) \models \theta_j$, and because $L(e) = t_j = \langle q, \theta_j, F_j, Q_j\rangle = \big\langle L(v_j), \theta_j, F_j, L(V_{j+1})\big\rangle$, the edge labelling is consistent.

- Since the target node set of the only edge starting from a node at level $0 \leq j < i + 1$ covers all nodes in V_{j+1}, it is clear that each node in V_j for some $1 \leq j < \omega$ is a successor of some node at level $j - 1$.

If $i < \omega$ (i.e., if $w \models (\psi_{\text{self-loop}} \mathsf{U_s}\, \psi_{\text{non-self-loop}})$), then the edge set E is finite. This implies that $\mathcal{E}(G) = \emptyset$, and thus G is trivially a partial fin-accepting run of $\mathcal{A}^q$ on w. Otherwise $\mathcal{E}(G)$ contains a unique sequence $r = (e_j)_{0 \leq j < \omega}$ of consecutive edges of E, where $e_j = \langle v_j, V_{j+1}\rangle$ for all $0 \leq j < \omega$.

Assume that r does not satisfy the fin-acceptance condition. Therefore, there exists a minimal index $0 \leq k < \omega$ 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 in $\mathcal{F}$ for all $k \leq j < \omega$. Let $f_{\min}$ be the minimal acceptance condition (under $\prec$) among these conditions. Because k is minimal, there exists another index $k \leq k' < \omega$ such that $f_{\min}$ is the minimal element of $\widetilde{F}_j$ for all $k' < j < \omega$.

Because $i = \omega$, $w \not\models (\psi_{\text{self-loop}} \mathsf{U_s}\, \psi_{\text{non-self-loop}})$. Therefore it is necessarily the case that $w \models \mathsf{G}\big(\psi_{\text{self-loop}} \wedge \bigwedge_{f \in \mathcal{F}} \mathsf{F}\psi_{\text{avoid}}(f)\big)$. Thus there exists an index $k' < l < \omega$ such that $w^l \models \psi_{\text{avoid}}(f_{\min}) \equiv \bigvee_{\substack{\langle q, \theta, F, Q'\rangle \in \Delta \\ q \in Q',\, f_{\min} \notin F}} \mu(\theta, Q')$. 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 $\widetilde{F}_l$, it follows that t_l will be chosen from the set T. But then $f_{\min}$ cannot be one of the acceptance conditions shared by all transitions t_j for all $k \leq j < \omega$, which is a contradiction. It follows that r satisfies the fin-acceptance condition, and G is a partial fin-accepting run of $\mathcal{A}^q$ on w.

Finally, let $v \in V_j$ for some $0 \leq j < \omega$ be a node with no outgoing edges in G. Then $v \neq v_0$, and thus it is one of the nodes $v_{j-1,k}$ for some $1 \leq k \leq n_{j-1}$ such that $L(v) = q_{j-1,k} \in Q_{j-1} \setminus \{q\}$. Because $w^{j-1} \models \mu(\theta_{j-1}, Q_{j-1})$, that is, $w^{j-1} \models \big(\theta_{j-1} \wedge \mathsf{X}(\bigwedge_{q' \in Q_{j-1} \setminus \{q\}} \varphi_{q'})\big)$, it follows that $w^j \models \varphi_{q'}$ for all $q' \in Q_{j-1} \setminus \{q\}$. In particular, $(\mathcal{A}^q)^{q_{j-1,k}}$ now has a fin-accepting run on w^j by assumption. Since v is an arbitrary node 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 2.3.10, 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 linear alternating automata, using Lemma 3.5.1 as a tool for proving the induction step.

Theorem 3.5.2 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton over the alphabet $\Sigma = 2^S$ for some finite set S. Then, 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 $w \models \varphi$.*

Proof: Because $\mathcal{A}$ is a linear 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 2.3.13, $\rho(q)$ can be defined as

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

where $Q_0 \stackrel{def}{=} \big\{ q \in Q \,\big|\, \text{for all } \langle q, \Gamma, F, Q' \rangle \in \Delta,\ Q' = \emptyset \text{ or } Q' = \{q\} \big\}$. We proceed by induction on $\rho(q)$. If $\rho(q) = 0$, then the result follows immediately for the subautomaton $\mathcal{A}^q$ by Lemma 3.5.1, since q has no successors different from itself (and thus the assumption needed in Lemma 3.5.1 holds trivially). Assume that the result holds for all subautomata $\mathcal{A}^q$, where $q \in Q$ satisfies $\rho(q) \le i$ for some $0 \le i < \omega$. Let $q' \in Q$ be a state for which $\rho(q') = i + 1$. Then, $\rho(q) \le i$ for all successors of q' excluding q' itself, and the result follows again for the subautomaton $\mathcal{A}^{q'}$ by Lemma 3.5.1 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 $\mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_I})$ (Proposition 2.3.11). $\square$

In principle, the proof of Theorem 3.5.2 gives an inductive procedure for finding an LTL formula that corresponds to a given linear alternating automaton by repeatedly using the formula given in Lemma 3.5.1 as a pattern for defining LTL formulas corresponding to states with increasing values of ρ. However, it is easy to see that the size of the formulas obtained from repeated applications of the translation pattern grows very rapidly as the value of ρ increases. We can give a rough indirect estimate on the maximal blow-up caused by this simple pattern by translating the formula obtained from the reverse translation back into an automaton and then using Corollary 3.3.2 to find an upper bound for its size. Ideally, because each of these transformations preserves the language of the automaton or the formula to which the transformation is applied, the size of the resulting automaton should not exceed the size of the original automaton. However, this is not the case with the reverse translation presented above.

Proposition 3.5.3 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton over the alphabet $\Sigma = 2^S$ for some finite set S, and let φ be the LTL formula obtained through a systematic application of the translation pattern given in Lemma 3.5.1 to $\mathcal{A}$ as described in Theorem 3.5.2. Then, the automaton $\mathcal{A}'$ obtained again from φ by the basic translation has at most $1 + |\Delta| + |Q|\big(|\mathcal{F}| + 2\big)$ states.*

Proof: By Theorem 3.5.2, the reverse translation pattern needs to be applied once to each state of $\mathcal{A}$. Since the size of $\mathcal{A}'$ depends on the number of distinct pure temporal subformulas of φ, we estimate the number of pure temporal subformulas generated in a single application of the reverse translation pattern; $|\mathrm{Temp}(\varphi)|$ will then not exceed the total number of pure temporal formulas generated over all applications of the pattern.

In the application of the reverse translation pattern to a state $q \in Q$, each occurrence of $\mu(\theta, Q')$ for some $\theta \in PL(S)$ and $Q' \subseteq Q$ will contribute a Next Time subformula to the result. From the definitions of $\psi_{\text{self-loop}}$, $\psi_{\text{non-self-loop}}$ and ψ_{avoid} it is easy to see that μ is always quantified over the transitions leaving the state q. We call the number of these transitions the *out-degree* of q and denote it by $\deg(q)$. The quantification over the acceptance sets $f \in \mathcal{F}$ contributes at most $|\mathcal{F}|$ distinct pure temporal F subformulas to the result. Furthermore, it is easy to see that the pattern also includes one G and one $\mathsf{U_s}$ formula as its subformulas. Thus, a single application of the pattern will give rise to at most $\deg(q) + |\mathcal{F}| + 2$ distinct pure temporal subformulas[2]. Quantifying over all states of the automaton, we find that φ will contain at most

$$\sum_{q \in Q} \big(\deg(q) + |\mathcal{F}| + 2\big) = \sum_{q \in Q} \deg(q) + \sum_{q \in Q} \big(|\mathcal{F}| + 2\big) = |\Delta| + |Q|\big(|\mathcal{F}| + 2\big)$$

distinct pure temporal subformulas. The result now follows directly by Corollary 3.3.2. $\square$

In addition to the linear dependency (in the number of acceptance conditions of the original automaton) on the size of the original automaton, the upper bound depends also explicitly on the number of transitions in the original automaton (which may be exponential in the size of the automaton). These dependencies clearly arise from the applications of the reverse translation pattern of Lemma 3.5.1, since the same dependencies already concern the number of temporal subformulas in the formula obtained using reverse translation. Because only a linear increase in the size of an alternating automaton may already result in an exponential worst-case increase in the size of a corresponding nondeterministic automaton (see Ch. 5), Proposition 3.5.3 provides evidence against the practicability of using the simple pattern in Lemma 3.5.1 as the sole basis for an efficient reverse translation procedure. Any practical implementation of such a procedure is therefore likely to be forced to tackle the problem of minimizing the blow-up in the size of intermediate results, for example, by using specially designed formulas for the reverse translation in different cases, or by applying aggressive simplification to each formula built in each reverse translation step.

[2]This number will not change when rewriting the F and G operators in terms of the more primitive operators before translating the formula back into an automaton.

4 IMPROVING THE TRANSLATION

In this chapter we explore methods for improving the basic translation presented in the previous chapter to generate smaller linear alternating automata. Minimization of automata built from LTL formulas has previously been studied both in the context of nondeterministic automata (e.g., [21, 22, 34, 62]) and alternating automata [25, 28, 58]; some of the proposed optimizations, such as formula-based simplifications [21, 62], apply independent of the type of the target automata. We shall extend the basic formula translation with some of these common optimizations and examine their special cases applicable to our linear alternating automata with acceptance conditions on transitions instead of states. We shall also investigate translation rule heuristics that allow simplification of the transition structure of the generated automata.

4.1 ELEMENTARY SIMPLIFICATION TECHNIQUES

4.1.1 Subformulas with Commutative Main Connectives

A well-known obvious heuristic for reducing the number of steps required for the automaton translation of a given formula φ is to order the top-level subformulas of each subformula of φ with a commutative binary main connective ($\vee$ or $\wedge$ in our set of basic operators) systematically (e.g., into some lexicographic order), for example, as an explicit preprocessing step before the translation. Even though this fixing of the order of subformulas has no effect on the worst-case size of the resulting automaton (by Corollary 3.3.2, this size does not depend on the subformulas of φ with a Boolean main connective), it may nevertheless help in reducing the number of translation steps, which depends on the number of syntactically distinct subformulas in φ. This way, we can easily avoid repeating potentially expensive automaton minimization operations (such as those presented later in this chapter) on isomorphic automata.

4.1.2 Transition Guard Simplification

As noted in Section 3.2, all transition guards of the automaton constructed using the translation rules are finite conjunctions of one or more atomic formulas (also known as *terms* in the literature [21]). Due to the associativity of the $\wedge$ operator, each guard formula θ can be written without parentheses as $\theta = \mu_1 \wedge \mu_2 \wedge \cdots \wedge \mu_n$, where each μ_i is an atomic formula for $1 \leq i \leq n$ $(1 \leq n < \omega)$. The formula can then be simplified using the commutativity of $\wedge$ and the classic identities of propositional logic:

$$\begin{array}{cc} (\theta \wedge \theta) \equiv \theta & (\theta \wedge \bot) \equiv (\bot \wedge \theta) \equiv \bot \\ (\theta \wedge \top) \equiv (\top \wedge \theta) \equiv \theta & (\theta \wedge \neg\theta) \equiv (\neg\theta \wedge \theta) \equiv \bot \end{array}$$

It is easy to see that any 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 $\mathcal{L}(\bot) = \emptyset$, no edge in a run of an alternating automaton on words over the alphabet 2^{AP} can be labelled with a transition having $\bot$ as its guard formula (i.e., there is no $\sigma \in 2^{AP} \cap \emptyset$ that could be used to label an edge of the run with such a transition consistently). Since the language inf- or fin-recognized by an alternating automaton depends only on the runs of the automaton, all transitions having an unsatisfiable guard can thus be removed from the automaton without changing its language.

A standard interpretation of the transition guards as Boolean functions over the atomic propositions allows the use of binary decision diagrams (BDDs) [5] for their representation [14, 43, 68]. Another simple alternative for manipulating guards in the above very restricted form $\theta = \mu_1 \wedge \mu_2 \wedge \cdots \wedge \mu_n$ (used, for example, by Gastin and Oddoux in their implementation [28]) is to represent each guard formula as a pair of sets of atomic propositions $\langle P_1, P_2 \rangle \in 2^{AP} \times 2^{AP}$, where P_1 (P_2) collects all atomic propositions $p \in AP$ for which $\mu_i = p$ ($\mu_i = \neg p$) for some $1 \leq i \leq n$; in this notation, the Boolean constant $\top$ corresponds to the pair $\langle \emptyset, \emptyset \rangle$. This representation 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 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 pairs of sets of atomic propositions corresponding to two guard formulas θ and θ', 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 $P_1 \cap P_2 \neq \emptyset$, and the propositional implication $\theta \rightarrow \theta'$ is valid iff $P_1' \subseteq P_1$ and $P_2' \subseteq P_2$.

4.1.3 Translation Example

We illustrate the basic translation and transition guard simplification with an example by translating the formula

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

into a linear alternating automaton. 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

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

This results in the formula

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

because this formula is in positive normal form, we can now start applying the translation rules. We first build automata shown in Fig. 4.1 (a) for the atomic subformulas of the formula $\varphi \stackrel{def}{\equiv} \Big(\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big) \wedge \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big) \Big)$. We 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. 4.1 (b). Because both of these subformulas are strong temporal eventualities, we associate a unique acceptance condition with each compound automaton. (In Figs. 4.1, 4.2 and 4.3, we omit the states not reachable from the initial states of the constructed automata, since they can always be removed by Proposition 2.3.11 without changing the language of the automaton.)

We then apply the $\mathsf{R_w}$ translation rule to the automata $\mathcal{A}_\perp$ and $\mathcal{A}_{(\top\,\mathsf{U_s}\,p_1)}$, and then to $\mathcal{A}_\perp$ and $\mathcal{A}_{(\top\,\mathsf{U_s}\,p_2)}$, to obtain the automata shown in Fig. 4.1 (c). Because $\mathsf{R_w}$ is a weak temporal eventuality, no new acceptance conditions are added to the automata.

Both automata in Fig. 4.1 (c) have transitions including $\perp$ as one of the conjuncts in their guards. All of these guards thus reduce to the unsatisfiable formula $\perp$, and the transitions can be removed from the automata as shown in Fig. 4.1 (d).

We next merge the automata built for the top-level subformulas of φ into the automaton shown in Fig. 4.1 (e) for the formula φ itself by using the $\wedge$ translation rule. Because of the intermediate simplification step (d), this automaton has four initial transitions; without the simplification, applying the $\wedge$ translation rule directly to the automata shown in Fig. 4.1 (c) would have resulted in an automaton with sixteen initial transitions. We can again simplify the guards of some transitions in this automaton to obtain the automaton shown in Fig. 4.1 (f).

The translation of the subformula $\psi \stackrel{def}{\equiv} \big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)$ proceeds similarly. We start from the automata built for the atomic subformulas (see Fig. 4.2 (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. 4.2 (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. 4.2 (c)).

We finally apply the $\vee$ translation rule to $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ to build the automaton shown in Fig. 4.3 for the formula

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

4.2 LANGUAGE CONTAINMENT CHECKING BETWEEN LINEAR ALTERNATING AUTOMATA

Before we discuss further simplification techniques for linear alternating automata, we review the problem of checking for *language containment* between linear alternating automata. Language containment provides a unified and powerful, yet conceptually simple, way to express conditions for the minimization of automata. We shall see such conditions applied to automata obtained during the formula translation throughout the rest of this chapter.

Clearly, for all alternating automata $\mathcal{A}_1$ and $\mathcal{A}_2$ having a common alphabet Σ, all words of the language $\mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ belong to the language $\mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ iff no word in $\mathcal{L}_{\text{fin}}(\mathcal{A}_1)$ belongs to the complement of $\mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ with respect to Σ^ω, i.e., $\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$ holds iff $\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \overline{\mathcal{L}_{\text{fin}}(\mathcal{A}_2)} = \emptyset$. This classic reformulation equates language containment with language emptiness and

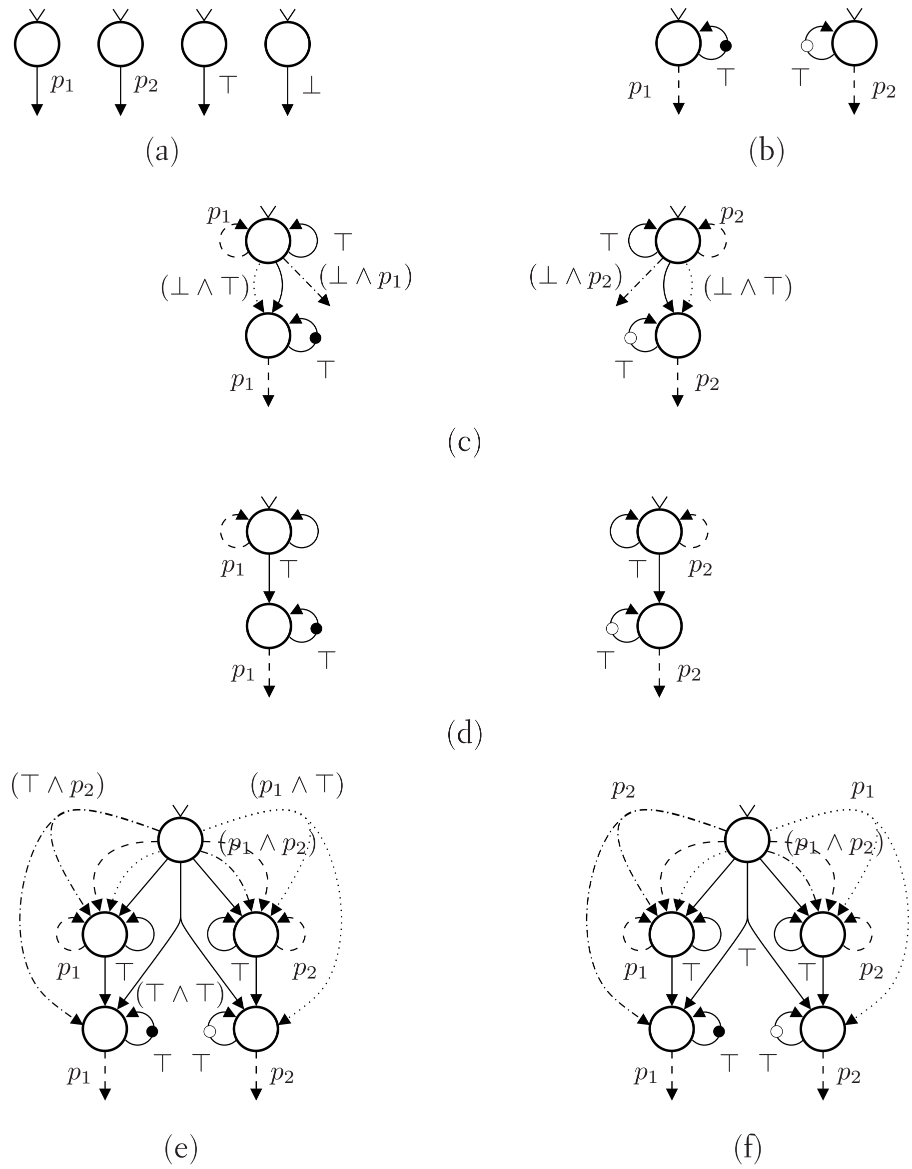
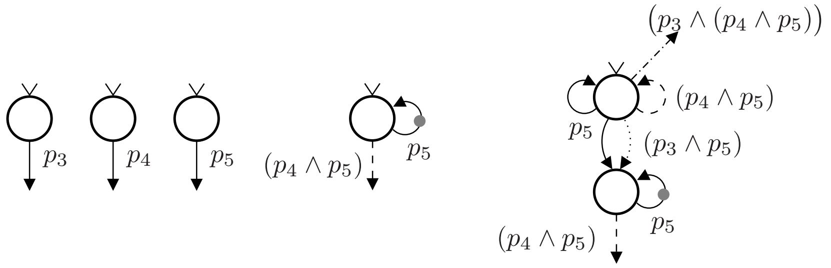


Fig. 4.1: Building an automaton for the LTL formula $\varphi \stackrel{def}{\equiv} \Big(\big(\bot \, \mathsf{R_w} \, (\top \, \mathsf{U_s} \, p_1) \big) \wedge \big(\bot \, \mathsf{R_w} \, (\top \, \mathsf{U_s} \, p_2) \big) \Big)$. (a) Automata for the atomic subformulas of φ; (b) Automata for the 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) The automata obtained from (c) by removing transitions with unsatisfiable guards; (e) Automaton for the formula φ; (f) The automaton obtained from (e) by transition guard simplification



(a) (b) (c)

Fig. 4.2: Building an automaton for the formula $\psi \overset{def}{\equiv} \big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)$. (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

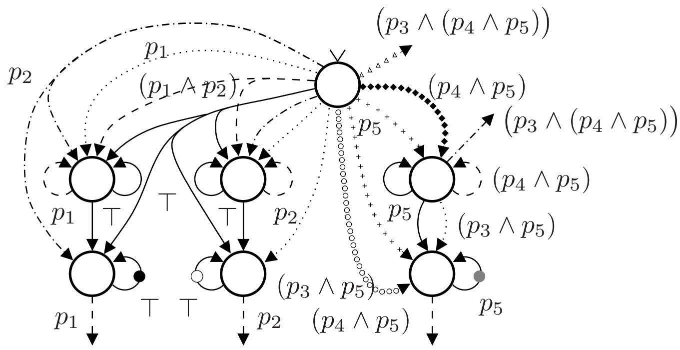


Fig. 4.3: Automaton for the LTL formula $\Big[\Big(\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big) \wedge \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big)\Big) \vee \big(p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5)\big)\Big]$

forms a first step towards handling language containment computationally. In particular, due to the interchangeability of linear alternating automata having the alphabet 2^S for some finite set S with formulas of the temporal logic $LTL(S)$, we can always represent both languages in the intersection as linear alternating automata: if $\varphi_1, \varphi_2 \in LTL(S)$ are two LTL formulas corresponding to the linear alternating automata $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, it is easy to see by Theorems 3.4.3 and 3.5.2 and the semantics of LTL that

$$\begin{aligned}
&\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \overline{\mathcal{L}_{\text{fin}}(\mathcal{A}_2)} \\
= {}&\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \overline{\mathcal{L}(\varphi_2)} \\
= {}&\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \big((2^S)^\omega \setminus \mathcal{L}(\varphi_2)\big) \\
= {}&\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}(\neg\varphi_2) \\
= {}&\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\text{fin}}(\overline{\mathcal{A}}_2) \text{ for an automaton } \overline{\mathcal{A}}_2 \text{ built for the LTL formula } \neg\varphi_2.
\end{aligned}$$

Language containment between $\mathcal{A}_1$ and $\mathcal{A}_2$ thus reduces to the emptiness of the intersection of languages of the automaton $\mathcal{A}_1$ and an automaton $\overline{\mathcal{A}}_2$ that can be built by using the formula translation procedure.

The ability to apply the basic translation for building the automata for the emptiness check requires knowledge on the formulas φ_1 and $\neg\varphi_2$. Because the language containment checks to be discussed often involve automata built during the translation of a formula φ into an automaton (i.e., automata built for the subformulas of φ), the bijective correspondence between these automata and the subformulas of φ ensures in many cases that the formulas φ_1 and $\neg\varphi_2$ are indeed known (or can be determined easily) before each language containment test. In this case we may also have the automata for the formulas φ_1 or $\neg\varphi_2$ already available; even if this is not the case, the automata can always be found by applying the basic translation to the positive normal forms of the formulas. (Because $|\text{Temp}(\varphi)| = |\text{Temp}(\neg\varphi)|$ holds for all LTL formulas φ, the automaton obtained this way for the language $\mathcal{L}(\neg\varphi_2)$ has at most $1 + |\text{Temp}(\varphi_2)|$ states by Corollary 3.3.2; thus, the size of this automaton does not exceed the size of the automaton $\mathcal{A}_2$ built from the formula φ_2.)

Nevertheless, plain formula translation is not by itself sufficient for handling language containment tests if we do not have the LTL formulas corresponding to the automata $\mathcal{A}_1$ or $\mathcal{A}_2$ available; in particular, in the case that φ_2 is in fact unknown, the above steps are not by themselves sufficient for finding an automaton for the language $\overline{\mathcal{L}_{\text{fin}}(\mathcal{A}_2)}$. In such a case, we may first have to apply, for example, the reverse translation to the automaton $\mathcal{A}_2$ to find an LTL formula to be negated and translated back into an automaton for the language $\overline{\mathcal{L}_{\text{fin}}(\mathcal{A}_2)}$. Alternatively, we could try to build the automaton $\overline{\mathcal{A}}_2$ from $\mathcal{A}_2$ by *dualization* [52] to find an automaton for this language directly, instead of making use of the two-way connection between LTL and linear alternating automata. However, direct complementation by dualization, for example, using the constructions of Vardi and Kupferman [41], or the related construction of Löding [46], is not possible using our definition of automata because of the generalized definition of acceptance and transition guards. In the following discussion, we shall therefore use the method based on reverse translation as a general complementation procedure for linear alternating automata. We shall nevertheless focus mostly on special cases that can be handled without such a procedure. Even though giving up the ability to complement arbitrary linear alternating automata (i.e., the ability to perform arbitrary language containment tests) will reduce the opportunities for the simplification of automata, we shall nevertheless obtain a collection of methods, all of which can be implemented by applying the basic translation procedure to formulas that are already known in the context.

Finally, we remark that language containment checking between linear alternating automata by reducing the problem to emptiness checking is likely to be feasible in practice only in simple cases, due to computational complexity issues. As a matter of fact, constructing the automata required for the language intersection emptiness test may already require exponential space (and hence at least exponential time) when using the chosen explicit representation for the transitions of alternating automata, even when assuming that explicit complementation of automata is not needed. For example, automata built from Boolean formulas $\bigwedge_{1 \leq i \leq n} \varphi_i$ (where $\varphi_i \equiv \bigvee_{1 \leq j \leq m_i} \psi_{i,j}$ for atomic formulas $\psi_{i,j}$, and $n, m_i \in \mathbb{N}$) in *conjunctive normal form* may have an exponential number of transitions in the size of the formula in the worst case. In fact, the language intersection emptiness problem for linear alternating automata is PSPACE-hard in general and can thus be expected to be PSPACE-complete at best: PSPACE-hardness can be shown via a reduction from the problem of LTL satisfiability, known to be PSPACE-complete [61] even for many restrictions of the logic [16]. More precisely, a formula $\varphi \in LTL(AP)$ is satisfiable iff $\mathcal{L}_{\text{fin}}(\mathcal{A}_\varphi) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}) \neq \emptyset$, where $\mathcal{A}_\varphi$ is an automaton built for the formula φ using a standard Boolean encoding (necessary for keeping the representation polynomial in the size of the formula) for the transition relation, and $\mathcal{A}$ is an automaton that fin-accepts the universal language $(2^{AP})^\omega$. Whether the problem of checking the emptiness of the language intersection is actually in PSPACE for generalized linear alternating automata is not entirely obvious without more detailed analysis; one possible way to establish this would be to try to find a polynomial-space reduction from the generalized definition to, for example, the definition of Vardi [73], for which the problem is known to be in PSPACE.

By the above observations, language containment checking should be applied with care in practice, using heuristics to estimate its feasibility in individual problem instances and limit the tests, for example, only to special cases of the theoretical constructions that follow in this chapter. We nevertheless state our results in terms of general language containment to allow for more flexibility in illustrating the intuition behind various simplification techniques.

4.3 HEURISTICS FOR TRANSLATION RULES

In this section we investigate methods for improving the basic formula translation by refining the translation rules defined in Sect. 3.2. We first review the conceptually simple idea of using language containment tests between component automata as "preprocessing steps" for the translation rules. A language containment relationship between two component automata may allow us to replace one of them with a simpler automaton before actually applying a translation rule. We shall also refine the translation rules themselves:

Table 4.1: 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
$(\varphi_1 \,\mathsf{U_s}\, \varphi_2)$	φ_2		$(\top \,\mathsf{U_s}\, \varphi_2)$	$(\varphi_2 \,\mathsf{R_s}\, \varphi_1)^\dagger$
$(\varphi_1 \,\mathsf{U_w}\, \varphi_2)$	φ_2		$\top$	$(\varphi_2 \,\mathsf{R_w}\, \varphi_1)^\dagger$
$(\varphi_1 \,\mathsf{R_s}\, \varphi_2)$	$(\varphi_2 \,\mathsf{U_s}\, \varphi_1)^\dagger$	$\bot$		φ_2
$(\varphi_1 \,\mathsf{R_w}\, \varphi_2)$	$(\varphi_2 \,\mathsf{U_w}\, \varphi_1)^\dagger$	$(\bot \,\mathsf{R_w}\, \varphi_2)$		φ_2

† See also Sect. 4.3.3.

instead of building a compound automaton from one or two component automata always in the same way, taking the properties of the component automata into account may in some cases allow the construction of compound automata with a simpler transition structure than the one determined by the basic rules.

4.3.1 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 simplification of the compound automaton built from these automata. For example, if the language of an LTL formula $\varphi_1 \in LTL(AP)$ is included in the language of another LTL formula $\varphi_2 \in LTL(AP)$ (i.e., if $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$), then $\mathcal{L}\big((\varphi_1 \wedge \varphi_2)\big) = \mathcal{L}(\varphi_1) \cap \mathcal{L}(\varphi_2) = \mathcal{L}(\varphi_1)$. Therefore we can reuse the automaton already built for φ_1 to obtain an automaton for $(\varphi_1 \wedge \varphi_2)$ directly, instead of 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 be obtained by using the translation rule: for example, there is no need to add a new initial state to the result.

Table 4.1 contains the formula simplification rules resulting from a number of possible assumptions about the relationship between the languages of two LTL formulas. Each cell of the table gives an LTL formula that is equivalent to the formula labelling the row of the cell under the language containment relationship given as the label of the cell's column; if the cell is empty, the assumption does not lead to direct simplification of the formula. The correctness of each rule can be verified directly using the semantics of LTL. Because the LTL formulas referred to in the conditions for language containment are subformulas of the compound formulas found as row labels in 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.

By the correspondence between LTL formulas and linear alternating automata, an automaton built for a formula in a (nonempty) cell of Table 4.1 can always be substituted for the compound automaton built for the formula labelling the row of the cell. The formula substitution is clearly preferable whenever either one of the original formula's subformulas can be discarded 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 subformulas reversed. We shall investigate these cases further in Sect. 4.3.3 where we use the same language containment assumptions to refine the translation rules for these connectives. The identities in Table 4.1 can nevertheless be used for reducing the effective number of distinct subformulas to which the translation needs to be applied when translating a formula φ into an automaton. For example, by treating all subformulas of the form $(\varphi_1 \,\mathsf{R_s}\, \varphi_2)$, where $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$, systematically as formulas of the form $(\varphi_2 \,\mathsf{U_s}\, \varphi_1)$, we may obviously be able to reuse an automaton built from $(\varphi_2 \,\mathsf{U_s}\, \varphi_1)$ to find an automaton for $(\varphi_1 \,\mathsf{R_s}\, \varphi_2)$. This effectively allows a reduction in $|\mathrm{Temp}(\varphi)|$ and therefore also in the worst-case size of an automaton for φ (Corollary 3.3.2). 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 4.1 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 full language containment is considered too expensive. Checking for syntactic special cases that imply language containment is a standard technique used in most actual implementations (e.g., [21, 62, 68]), usually as a preprocessing step; clearly, performing the translation incrementally supports combining the formula rewriting step easily also 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 ($\mathcal{L}(\varphi) \subseteq \mathcal{L}(\bot)$), 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 improving the opportunities for Boolean constant propagation at the cost of an increase in the number of language containment tests.

4.3.2 Modified Translation Rules: The $\wedge$ Connective

By the definition of the translation rule given for the $\wedge$ connective, if either of the component automata to which the rule is applied during translation has an initial self-loop transition, the rule "unrolls" this self-loop by adding to the constructed automaton an initial transition that includes the initial state of the component automaton in its target states (cf. Fig. 3.1, page 31). This

Fig. 4.4: Two automata for the LTL formula $(\mathsf{G}p_1 \wedge \mathsf{G}p_2)$. (a) Automaton built for the formula using the translation rules (where transitions with unsatisfiable guards have been removed); (b) Minimal automaton for the formula

implies that the initial state of the component automaton will still remain in the final automaton at the end of translation. Thus, for example, using the translation rules for defining an automaton for the LTL formula

$$(\mathsf{G}p_1 \wedge \mathsf{G}p_2) \equiv \big((\bot \,\mathsf{R_s}\, p_1) \wedge (\bot \,\mathsf{R_s}\, p_2)\big) \in LTL\big(\{p_1, p_2\}\big)$$

results, even after removing transitions with unsatisfiable guards as described in Sect. 4.1.2, in a three-state automaton that contains the initial states of the automata built for the formulas $(\bot \,\mathsf{R_s}\, p_1)$ and $(\bot \,\mathsf{R_s}\, p_2)$; see Fig. 4.4 (a) for illustration. This is obviously not the minimal result, since the language of $(\mathsf{G}p_1 \wedge \mathsf{G}p_2)$ is clearly fin-recognized also by the single-state automaton shown in Fig. 4.4 (b). In this section, we refine the translation rule given for the $\wedge$ connective to improve the translation in simple cases analogous to the above example.

Consider two linear alternating automata $\mathcal{A}_1$ and $\mathcal{A}_2$, both of which fin-accept a word w. Assume that the automata have fin-accepting runs on w that include a nonempty sequence of edges corresponding to self-loops of the respective automata, possibly followed by an edge that is not labelled with a self-loop (cf. Proposition 2.3.7). If $\mathcal{A}_1$ and $\mathcal{A}_2$ are used as component automata in an application of the translation rule given for the $\wedge$ connective, then the automaton $\mathcal{A}$ built by the rule simulates the synchronous behavior of $\mathcal{A}_1$ and $\mathcal{A}_2$ on the same input (see the discussion in Sect. 3.2). Informally, when $\mathcal{A}$ takes its first transition in a fin-accepting run, it reads the first symbol of the input and spawns a copy of each component automaton, which then work in parallel on the rest of the input. Therefore, the run of $\mathcal{A}$ separates into two (not necessarily disjoint) “branches” corresponding to the fin-accepting runs of the component automata. Since both of these branches begin with sequences of initial self-loops of $\mathcal{A}_1$ and $\mathcal{A}_2$, the run of $\mathcal{A}$ could be simplified by merging the initial self-loop parts of these branches together, and thus, in effect, postpone the branch separation beyond the first transition step as shown in Fig. 4.5. Furthermore, if either $\mathcal{A}_1$ or $\mathcal{A}_2$ stops looping through its initial state after the branch separation, $\mathcal{A}$ will not have to spawn both of these component automata, but only their subautomata. If $\mathcal{A}$ can in a similar way avoid spawning $\mathcal{A}_1$ or $\mathcal{A}_2$ for all pairs of fin-accepting runs of the component automata on all inputs, it follows that $\mathcal{A}$ does not need any initial transitions to one or both of the initial states of these automata. Unlike in the basic construction, the initial state of at least one component automaton now remains a possible candidate for eventual removal from the

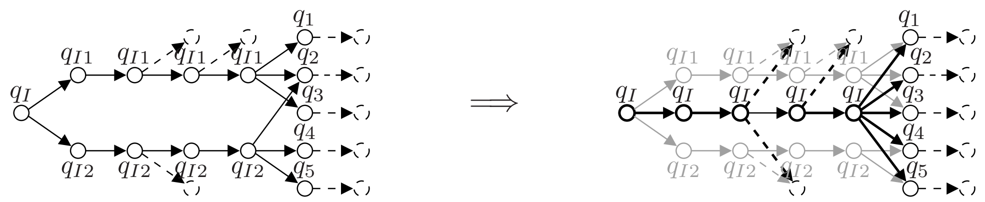


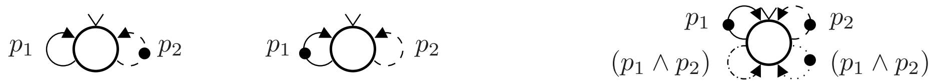
Fig. 4.5: Merging the initial parts of two branches (runs of subautomata with initial states q_{I1} and q_{I2}) in a run of an automaton having initial state q_I

automaton obtained at the end of the formula translation, provided that no other translation rule creates transitions to this state, either.

The above strategy of merging the initial parts 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. Similar to the basic construction, we first collect all pairs of initial transitions of the component automata and use the basic construction for all transition pairs, either element 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 synchronizing transitions; however, since the intention is to obtain a self-loop that prevents the run of $\mathcal{A}$ from separating into two branches, 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 would seem reasonable to try a similar strategy by taking the union of the acceptance conditions of the pair of synchronizing transitions to prevent $\mathcal{A}$ from possibly fin-recognizing more inputs than its component automata. Indeed, this proves to be sufficient if the component automata do not share any common acceptance conditions. However, as illustrated by the following example, this simple strategy may lead to incorrect results if the sets of acceptance conditions of the component automata are not disjoint.

Example 4.3.1 Figure 4.6 shows two alternating automata having the alphabet $2^{\{p_1,p_2\}}$, together with a linear alternating automaton (with transition guards simplified as described in Sect. 4.1) obtained by the above informal construction for merging the initial self-loops of the two automata into initial self-loops of the third automaton. It is easy to see that both automata in Fig. 4.6 (a) fin-accept the word $\big(\{p_1\}\{p_2\}\big)^\omega$, but this word does not belong to the language of the automaton (b): on this input, this automaton can only take transitions, all of which include the same acceptance condition. ■

The problem in the previous example arises because the component automata share a common acceptance condition that is included in the acceptance conditions of only one of two self-loops merged together. By adding a requirement that permits the merging of two self-loops only if they share a common subset of the acceptance conditions shared by both automata, we can prove the main result of this section. (In the above discussion, we



(a) (b)

Fig. 4.6: Self-loop pairs of automata that share common acceptance conditions cannot always be merged into single self-loops by forming their acceptance conditions as the union of the acceptance conditions associated with the merged transitions. (a) Two automata sharing a common acceptance condition; (b) Automaton obtained from the two automata by merging pairs of their initial transitions with their acceptance conditions

required that both transitions to be merged are initial *self-loops* of the component automata. We actually prove a slightly more general result: the transitions can be merged if the initial states of the component automata are included in the union of their target state sets.)

Proposition 4.3.2 *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 linear alternating automata ($q_{I1} \neq q_{I2}$) such that $\mathcal{A}_1^q = \mathcal{A}_2^q$ holds for all $q \in Q_1 \cap Q_2$. Assume also that $\mathcal{A}_1$ and $\mathcal{A}_2$ have been simplified in the sense of Corollary 2.3.17, that is, the set of acceptance conditions of each non-self-loop transition of $\mathcal{A}_1$ or $\mathcal{A}_2$ is empty. Define the linear alternating automaton $\mathcal{A} = \langle 2^{AP}, Q, \Delta, q_I, \mathcal{F}\rangle$, where $Q \stackrel{def}{=} Q_1 \cup Q_2 \cup \{q_I\}$ ($q_I \notin Q_1 \cup Q_2$),*

$$\Delta \stackrel{def}{=} \Delta_1 \cup \Delta_2 \cup \left\{ \big\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q'_1 \cup Q'_2 \big\rangle \;\middle|\; \begin{array}{l} \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2, \\ \{q_{I1}, q_{I2}\} \not\subseteq Q'_1 \cup Q'_2 \text{ or} \\ \big[q_{I1} \in Q'_1, q_{I2} \in Q'_2, \\ \quad F_1 \cap \mathcal{F}_1 \cap \mathcal{F}_2 \neq F_2 \cap \mathcal{F}_1 \cap \mathcal{F}_2\big] \end{array} \right\}$$
$$\cup \left\{ \begin{array}{l} \big\langle q_I, (\theta_1 \wedge \theta_2), F_1 \cup F_2, \\ \quad (Q'_1 \cup Q'_2 \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\}\big\rangle \end{array} \;\middle|\; \begin{array}{l} \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1, \\ \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2, \\ \{q_{I1}, q_{I2}\} \subseteq Q'_1 \cup Q'_2, \text{ and} \\ \text{if } q_{I1} \in Q'_1, q_{I2} \in Q'_2, \\ \text{then } F_1 \cap \mathcal{F}_1 \cap \mathcal{F}_2 \\ \qquad = F_2 \cap \mathcal{F}_1 \cap \mathcal{F}_2 \end{array} \right\}$$

and $\mathcal{F} \stackrel{def}{=} \mathcal{F}_1 \cup \mathcal{F}_2$. Then, for all $w \in (2^{AP})^\omega$, $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$ iff $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$.

Proof: ("$\Rightarrow$") Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$. 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 sequence of consecutive 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 if $i < \omega$, then $L(e_i)$ is an initial transition of $\mathcal{A}$ that is not a self-loop.

Let $L(e_j) = t_j = \langle q_I, \theta_j, F_j, Q'_j\rangle$ for all $0 \leq j < i + 1$. Let $0 \leq j < i + 1$. From the definition of $\mathcal{A}$ it follows that t_j corresponds to a pair of transitions $t_{j,1} = \langle q_{I1}, \theta_{j,1}, F_{j,1}, Q'_{j,1}\rangle \in \Delta_1$ and $t_{j,2} = \langle q_{I2}, \theta_{j,2}, F_{j,2}, Q'_{j,2}\rangle \in \Delta_2$ for some $\theta_{j,1}, \theta_{j,2} \in PL(AP)$, $F_{j,1} \subseteq \mathcal{F}_1$, $F_{j,2} \subseteq \mathcal{F}_2$, $Q'_{j,1} \subseteq Q_1$ and $Q'_{j,2} \subseteq Q_2$ such that $\theta_j = (\theta_{j,1} \wedge \theta_{j,2})$. 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 $0 \le j < i$ 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 $0 \le n_{j,k} < \omega$. If $i < \omega$, write $Q'_{i,k} = \{q_{i,k,1}, q_{i,k,2}, \ldots, q_{i,k,n_{i,k}}\}$ for some $0 \le n_{i,k} < \omega$ ($k \in \{1,2\}$).

We build a partial fin-accepting run $G' = \langle V', E', L' \rangle$ of $\mathcal{A}_1$ on w. We define the levels of G' inductively: let $V'_0 \stackrel{\mathit{def}}{=} \{v_{0,1}\}$ and $L'(v_{0,1}) \stackrel{\mathit{def}}{=} q_{I1}$, and assume that V'_j and $L'(v)$ have already been defined for some $0 \le j < i+1$ and for all $v \in V'_j$, respectively. For all $v \in V'_j$, let $S(v) \stackrel{\mathit{def}}{=} \emptyset$ if $L'(v) \notin \{q_{I1}, q_{I2}\}$. Otherwise, if $L'(v) = q_{Ik}$ for some $k \in \{1,2\}$, let

$$S(v) \stackrel{\mathit{def}}{=} \{v_{j,k,1}, \ldots, v_{j,k,n_{j,k}}\} \cup \begin{cases} \{v_{j+1,l} \,\big|\, l \in \{1,2\}, q_{Il} \in Q'_{j,k}\} & \text{if } j < i \\ \emptyset & \text{if } i < \omega,\, j = i. \end{cases}$$

Let $V'_{j+1} \stackrel{\mathit{def}}{=} \bigcup_{v \in V'_j} S(v)$, and define the labelling for the nodes at level $j+1$ by setting

$$L'(v_{j,k,l}) \stackrel{\mathit{def}}{=} q_{j,k,l} \qquad \text{and} \qquad L'(v_{j+1,k}) \stackrel{\mathit{def}}{=} q_{Ik}$$

for all (relevant) combinations of $k \in \{1,2\}$ and $1 \le l \le n_{j,k}$. If $i < \omega$, let $V'_j \stackrel{\mathit{def}}{=} \emptyset$ for all $i+1 < j < \omega$. To complete the definition of G', let $E' \stackrel{\mathit{def}}{=} \big\{ \langle v_{j,k}, S(v_{j,k}) \rangle \,\big|\, v_{j,k} \in V'_j \text{ for some } 0 \le j < i+1 \text{ and } k \in \{1,2\} \big\}$, and for all $e = \big\langle v_{j,k}, S(v_{j,k}) \big\rangle \in E'$, let $L'(e) \stackrel{\mathit{def}}{=} t_{j,k}$.

We check that G' is a partial run of $\mathcal{A}_1$ on w. First, we see that $V_0 = \{v_{0,1}\}$, $L'(v_{0,1}) = q_{I1}$, V' is partitioned into finite disjoint levels, and all edges of E' are between successive levels of G'.

Let $v \in V'_j$ for some $0 \le j < \omega$. Then $v = v_{j-1,k,l}$ for some $k \in \{1,2\}$ and $1 \le l \le n_{j-1,k}$, or $v = v_{j,k}$ for some $k \in \{1,2\}$. In the former case, v has no outgoing edges and satisfies the requirements of a partial run of $\mathcal{A}_1$ trivially. Otherwise $v = v_{j,k}$ has the unique outgoing edge $\big\langle v, S(v) \big\rangle \in E'$ labelled 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 $e_j \in V_j \times 2^{V_{j+1}} \subseteq E$ is labelled in G with the transition $t_j = \langle q_I, \theta_j, F_j, Q'_j \rangle \in \Delta$, $w(j) \models \theta_j$ (G is a run of $\mathcal{A}$). Because $\theta_j = (\theta_{j,1} \wedge \theta_{j,2})$, it follows that $w(j) \models \theta_{j,k}$. Moreover, $L'(v_{j,k}) = q_{Ik}$, and it is straightforward to check from the definitions that also $L'\big(S(v)\big) = Q'_{j,k}$ holds. Thus the edge labelling of G' is consistent.

If $v' \in V'_j$ for some $1 \le j < \omega$, then, 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 $v' \in S(v)$. Because $j - 1 < i + 1$, $L'(v) \in \{q_{I1}, q_{I2}\}$ implies that $v = v_{j-1,k}$ for some $k \in \{1,2\}$. Because $\big\langle v_{j-1,k}, S(v_{j-1,k}) \big\rangle \in E'$, it follows that v' is a successor of the node v at level $j-1$. This shows that G' is a partial run of $\mathcal{A}_1$ on w.

If $V'_j = \emptyset$ for some $0 \le j < \omega$ (which always holds if $i < \omega$), then E' is finite. In this case $\mathcal{E}(G') = \emptyset$, and G' is a partial fin-accepting run of $\mathcal{A}$ on w. Otherwise $i = \omega$, and G' contains an infinite sequence of consecutive edges $r = (e'_j)_{0 \le j < \omega}$, where $e'_j \in E' \cap (V'_j \times 2^{V'_{j+1}})$ for all $0 \le j < \omega$. By the definition of E', $e'_j = \big\langle v_{j,k}, S(v_{j,k}) \big\rangle$ for some $k \in \{1,2\}$, and thus $L'(e'_j) \in \{t_{j,1}, t_{j,2}\}$. Suppose that $\mathrm{fin}(r) \neq \emptyset$. Then there exists an acceptance condition $f \in \mathcal{F}_1 \cup \mathcal{F}_2$ and an index $0 \le l < \omega$ such that for all $l \le j < \omega$, $L'(e'_j)$ includes f in its acceptance conditions. Because $i = \omega$, each edge e_j in the run G is labelled with a self-loop of $\mathcal{A}$ for all $0 \leq j < \omega$, and thus $L(e_j)$ always includes the acceptance conditions of both $t_{j,1}$ and $t_{j,2}$ in its acceptance conditions. But then also $\mathrm{fin}\big((e_j)_{0\leq j<\omega}\big) \neq \emptyset$, which contradicts the assumption that G is a fin-accepting run of $\mathcal{A}$. Therefore $\mathrm{fin}(r)$ is necessarily empty, and G' is a partial fin-accepting run of $\mathcal{A}_1$ on w.

Let $v \in V'_j$ for some $0 \leq j < \omega$ be a node with no outgoing edges in G'. Then v is one of the nodes $v_{j-1,k,l}$ for some $k \in \{1,2\}$ and $1 \leq l \leq n_{j-1,k}$, and $L'(v) \in Q'' \stackrel{def}{=} \{q_{j-1,k,1},\ldots,q_{j-1,k,n_{j-1,k}}\} \subseteq Q_1$ (because G' is a run of $\mathcal{A}_1$, all nodes in G' are labelled with descendants of $L(v_{0,1}) = q_{I1}$ by Proposition 2.3.6). In the original run G, $e_{j-1} \in V_{j-1} \times 2^{V_j}$, and $L(e_{j-1}) = \langle q_I, \theta_{j-1}, F_{j-1}, Q'_{j-1}\rangle$, where $Q'' \subseteq Q'_{j-1}$. Because G is a fin-accepting run of $\mathcal{A}$ on w, $\mathcal{A}^q$ fin-accepts w^j for all $q \in Q'_{j-1}$, and because $Q'' \subseteq Q_1$, $\mathcal{A}^q = \mathcal{A}_1^q$ for all $q \in Q''$. Thus especially $\mathcal{A}_1^{L'(v)}$ has a fin-accepting run on w^j, and because this result holds for all $v \in V'$, we conclude that G' can be extended into a full fin-accepting run of $\mathcal{A}_1$ on w by Proposition 2.3.10.

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 $V'_0 \stackrel{def}{=} \{v_{0,2}\}$ with $L'(v_{0,2}) \stackrel{def}{=} q_{I2}$), we obtain a fin-accepting run of $\mathcal{A}_2$ on w. Thus $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)$ and $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$, and the result follows.

("$\Leftarrow$") Let $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1) \cap \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2)$, and let $G^1 = \langle V^1, E^1, L^1\rangle$ and $G^2 = \langle V^2, E^2, L^2\rangle$ be fin-accepting runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ on w, respectively. (Without loss of generality, we may assume that $V^1 \cap V^2 = E^1 \cap E^2 = \emptyset$.) By Proposition 2.3.7, there exist two indices $0 \leq i_1 \leq \omega$ and $0 \leq i_2 \leq \omega$ such that for all $k \in \{1,2\}$, G^k contains a sequence of consecutive edges $(e_{j,k})_{0\leq j<i_k+1}$, $e_{j,k} \in E^k \cap (V_j^k \times 2^{V_{j+1}^k})$, such that for all $0 \leq j < i_k$, $L^k(e_{j,k}) \in \Delta_k$ is an initial self-loop of $\mathcal{A}_k$, and if $i_k < \omega$, then $L^k(e_{i_k,k}) \in \Delta_k$ is an initial transition of $\mathcal{A}_k$ that is not a self-loop.

Our intention is to use the sequences $(e_{j,1})_{0\leq j<i_1+1}$ and $(e_{j,2})_{0\leq j<i_2+1}$ to construct a partial fin-accepting run for the compound automaton $\mathcal{A}$ built from $\mathcal{A}_1$ and $\mathcal{A}_2$. This partial run will again consist of a sequence of edges labelled with initial self-loops of $\mathcal{A}$, possibly followed by an edge labelled with an initial transition that is not a self-loop of $\mathcal{A}$. For this purpose, we need the following result.

Claim. *There exists an index* $0 \leq i \leq \omega$ *and, for each* $k \in \{1,2\}$, *a sequence* $(t_{j,k})_{0\leq j<i+1}$ *of initial transitions of* $\mathcal{A}_k$ *(where* $t_{j,k} = \langle q_{Ik}, \theta_{j,k}, F_{j,k}, Q'_{j,k}\rangle \in \Delta_k$ *for all* $0 \leq j < i+1$ *and* $k \in \{1,2\}$*) such that* $w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$ *for all* $0 \leq j < i+1$, $\{q_{I1}, q_{I2}\} \subseteq Q'_{j,1} \cup Q'_{j,2}$ *for all* $0 \leq j < i$, *and one of the following conditions holds:*

(a) $i < \omega$, *and* $\{q_{I1}, q_{I2}\} \not\subseteq Q'_{i,1} \cup Q'_{i,2}$,

(b) $i = \omega$, *and there exists an index* $0 \leq i' < \omega$ *such that* $t_{j,k}$ *is an initial self-loop of* $\mathcal{A}_k$ *for all* $i' \leq j < \omega$ *and* $k \in \{1,2\}$*; or*

(c) $i = \omega$, $t_{j,k}$ *is an initial self-loop of* $\mathcal{A}_k$ *for some* $k \in \{1,2\}$ *for all* $0 \leq j < \omega$, *and* $t_{j,3-k}$ *is an initial transition of* $\mathcal{A}_{3-k}$ *for all* $0 \leq j < \omega$ *such that* $t_{j,3-k}$ *is not a self-loop of* $\mathcal{A}_{3-k}$ *for infinitely many* j.

Proof: If $i_1 = i_2 < \omega$, then, by the choice of i_1 and i_2, q_{Ik} is not included in the target states of $L^k(e_{i_k,k})$ for $k \in \{1,2\}$. Assume nevertheless that q_{I1} and q_{I2} are both included in the union of the target states of these transitions. Therefore it must be case that q_{I1} is a target state of $L^2(e_{i_2,2}) \in \Delta_2$, and q_{I2} is a target state of $L^1(e_{i_1,1}) \in \Delta_1$, and thus $\{q_{I1}, q_{I2}\} \subseteq Q_1 \cap Q_2$. Because $\mathcal{A}_1^q = \mathcal{A}_2^q$ holds for all $q \in Q_1 \cap Q_2$, it now follows that both $\mathcal{A}_1$ and $\mathcal{A}_2$ contain a cycle that is not a self-loop. However, this contradicts the linearity of $\mathcal{A}_1$ and $\mathcal{A}_2$. Therefore either q_{I1} or q_{I2} must be missing from the union of the target states of $L^1(e_{i_1,1})$ and $L^2(e_{i_2,2})$. We can now choose $i \overset{def}{=} i_1 = i_2$ and define $t_{j,k} \overset{def}{=} L^k(e_{j,k}) = \langle q_{Ik}, \theta_{j,k}, F_{j,k}, Q'_{j,k}\rangle$ for all $0 \le j \le i$ and $k \in \{1,2\}$. Because G^1 and G^2 are runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ on w, it is easy to see that $w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$ holds for all $0 \le j \le i$. Thus the transition sequences $(t_{j,1})_{0\le j\le i}$ and $(t_{j,2})_{0\le j\le i}$ satisfy the claim such that condition (a) holds.

If $i_1 = i_2 = \omega$, then $L^k(e_{j,k})$ is an initial self-loop of $\mathcal{A}_k$ for all $k \in \{1,2\}$ and $0 \le j < \omega$. By defining $t_{j,k} \overset{def}{=} L^k(e_{j,k})$ for all $0 \le j < \omega$ and $k \in \{1,2\}$, we see that $\{q_{I1}, q_{I2}\} \subseteq Q'_{j,1} \cup Q'_{j,2}$ and $w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$ hold for all $0 \le j < \omega$. Thus the claim (case (b)) is now satisfied with $i \overset{def}{=} \omega$ and $i' \overset{def}{=} 0$.

If $i_1 \ne i_2$, then $i_k < i_{\bar{k}}$ for some $k \in \{1,2\}$, where $\bar{k} \overset{def}{=} 3 - k$, and $i_k < \omega$. Similar to above, we can define $t_{j,k} \overset{def}{=} L^k(e_{j,k})$ for all $0 \le j \le i_k$ and $k \in \{1,2\}$; then $\{q_{I1}, q_{I2}\} \subseteq Q'_{j,1} \cup Q'_{j,2}$ holds for all $0 \le j < i_k$, and $w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$ holds for all $0 \le j \le i_k$.

By the choice of i_k, $q_{Ik} \notin Q'_{i_k,k}$. If $q_{Ik} \notin Q'_{i_k,\bar{k}}$ also holds, then it is easy to see that the transition sequence defined above satisfies the conditions (case (a)) in the claim with $i \overset{def}{=} i_k$.

Otherwise, if $q_{Ik} \in Q'_{i_k,\bar{k}}$, then $\{q_{I1}, q_{I2}\} \subseteq Q'_{i_k,1} \cup Q'_{i_k,2}$, because $L^{\bar{k}}(e_{i_k,\bar{k}})$ is a self-loop of $\mathcal{A}_{\bar{k}}$ by the choice of $i_{\bar{k}}$ and the fact that $i_k < i_{\bar{k}}$. It follows that $e_{i_k,\bar{k}}$ has a target node (at level $i_k + 1$ in $G^{\bar{k}}$) labelled with q_{Ik}, and thus $q_{Ik} \in Q_1 \cap Q_2$. By Proposition 2.3.9, this implies that $\mathcal{A}_{\bar{k}}^{q_{Ik}}$ $(= \mathcal{A}_k^{q_{Ik}})$ fin-accepts w^{i_k+1}, and thus we can extract from $G^{\bar{k}}$ a fin-accepting run of $\mathcal{A}_k$ on w^{i_k+1}. From Proposition 2.3.7 it now follows that there exists an index $i_k + 1 \le i'_k < \omega$ such that the original edge sequence $(e_{j,k})_{0\le j\le i_k}$ can be extended with another edge sequence $(e_{j,k})_{i_k+1\le j<i'_k+1}$, $e_{j,k} \in E^{\bar{k}} \cap (V_j^{\bar{k}} \times 2^{V_{j+1}^{\bar{k}}})$, such that for all $i_k + 1 \le j < i'_k$, $L^{\bar{k}}(e_{j,k}) \in \Delta_k$ is an initial self-loop of $\mathcal{A}_k$, and if $i'_k < \omega$, then $L^{\bar{k}}(e_{i'_k,k}) \in \Delta_k$ is an initial non-self-loop transition of $\mathcal{A}_k$.

If i'_k is still strictly less than $i_{\bar{k}}$, we can apply the above construction repeatedly to extend the edge sequence $(e_{j,k})_{0\le j<i'_k+1}$ with segments of consecutive edges in $G^{\bar{k}}$ (corresponding to path fragments of separate runs of $\mathcal{A}_k$ embedded in $G^{\bar{k}}$) labelled (in $G^{\bar{k}}$) with initial transitions of $\mathcal{A}_k$ until we either find a (finite) index $0 \le i'' < i_{\bar{k}}$ such that either q_{I1} or q_{I2} is missing from the union of the target states of $L^{\bar{k}}(e_{i'',1})$ and $L^{\bar{k}}(e_{i'',2})$, or the repeated extension of $(e_{j,k})_{0\le j<i_k+1}$ results in an edge sequence $(e_{j,k})_{0\le j<i''+1}$ for some $0 \le i'' \le \omega$ such that $i_{\bar{k}} \le i''$. We can now continue the definition of the transition sequences by letting

$t_{j,\bar{k}} \stackrel{def}{=} L^{\bar{k}}(e_{j,\bar{k}})$ and $t_{j,k} \stackrel{def}{=} L^{\bar{k}}(e_{j,k})$ for all $i_k + 1 \le j < \min\{i'', i_{\bar{k}}\} + 1$; by the construction, $\{q_{I1}, q_{I2}\} \subseteq Q'_{j,1} \cup Q'_{j,2}$ then holds for all $0 \le j < \min\{i'', i_{\bar{k}}\}$, and $w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$ holds for all $0 \le j < \min\{i'', i_{\bar{k}}\} + 1$.

If $i'' < i_{\bar{k}}$ or $i'' = i_{\bar{k}} < \omega$, then $\{q_{I1}, q_{I2}\} \not\subseteq Q'_{i'',1} \cup Q'_{i'',2}$, and case (a) of the claim is satisfied with $i \stackrel{def}{=} i''$. This case of the claim also holds (with $i \stackrel{def}{=} i_{\bar{k}}$) if $i_{\bar{k}} < i''$: clearly, $q_{I\bar{k}} \notin Q'_{i_{\bar{k}},\bar{k}}$ holds by the choice of $i_{\bar{k}}$, and $q_{I\bar{k}} \notin Q'_{i_{\bar{k}},k}$ must hold also, because $\mathcal{A}_{3-k}$ is a linear alternating automaton. (Because $q_{Ik} \in Q'_{i_k,\bar{k}}$, it is a target state of an initial transition of $\mathcal{A}_{3-k}$, and thus no initial transition of $\mathcal{A}_k$ can have $q_{I\bar{k}}$ as its target state, since otherwise $\mathcal{A}_{3-k}$ would contain a loop that is not a self-loop.)

In the remaining case, $i'' = i_{\bar{k}} = \omega$. By choosing $i \stackrel{def}{=} \omega$, case (b) or (c) of the claim is necessarily satisfied. ■

Let $(t_{j,1})_{0\le j<i+1} = \big(\langle q_{I1}, \theta_{j,1}, F_{j,1}, Q'_{j,1}\rangle\big)_{0\le j<i+1} \in \Delta_1^{i+1}$ and $(t_{j,2})_{0\le j<i+1} = \big(\langle q_{I2}, \theta_{j,2}, F_{j,2}, Q'_{j,2}\rangle\big)_{0\le j<i+1} \in \Delta_2^{i+1}$ for some $0 \le i < \omega$ be two sequences of initial transitions of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, such that the above claim is satisfied. Let $\hat{\imath} \stackrel{def}{=} \min\big(\{i\} \cup \{0 \le j < i \mid q_{I1} \in Q'_{j,1}, q_{I2} \in Q'_{j,2}, F_{j,1} \cap \mathcal{F}_1 \cap \mathcal{F}_2 \ne F_{j,2} \cap \mathcal{F}_1 \cap \mathcal{F}_2\}\big)$. For all $0 \le j < \hat{\imath}$, write $Q'_{j,k} \setminus \{q_{I1}, q_{I2}\} = \{q_{j,k,1}, \ldots, q_{j,k,n_{j,k}}\}$ for some $0 \le n_{j,k} < \omega$, and if $\hat{\imath} < \omega$, write $Q'_{\hat{\imath},k} = \{q_{\hat{\imath},k,1}, \ldots, q_{\hat{\imath},k,n_{\hat{\imath},k}}\}$ for some $0 \le n_{\hat{\imath},k} < \omega$. Define the graph $G = \langle V, E, L\rangle$, where

- $V_0 \stackrel{def}{=} \{v_0\}$,
  $V_{j+1} \stackrel{def}{=} \{v_{j,1,1}, \ldots, v_{j,1,n_{j,1}}\} \cup \{v_{j,2,1}, \ldots, v_{j,2,n_{j,2}}\} \cup \begin{cases} \{v_{j+1}\} & \text{if } j < \hat{\imath} \\ \emptyset & \text{otherwise} \end{cases}$
  for all $0 \le j < \hat{\imath} + 1$, and if $\hat{\imath} < \omega$, let $V_j \stackrel{def}{=} \emptyset$ for all $\hat{\imath} + 1 < j < \omega$;
- $E \stackrel{def}{=} \bigcup_{0\le j<\hat{\imath}+1} \big\{\langle v_j, V_{j+1}\rangle\big\}$; and
- for all $0 \le j < \hat{\imath} + 1$, let $L(v_j) \stackrel{def}{=} q_I$ and $L(v_{j,k,l}) \stackrel{def}{=} q_{j,k,l}$ for all $k \in \{1, 2\}$ and $1 \le l \le n_{j,k}$. Furthermore, let $L\big(\langle v_j, V_{j+1}\rangle\big) \stackrel{def}{=} \big\langle q_I, (\theta_{j,1} \wedge \theta_{j,2}), F_{j,1} \cup F_{j,2}, (Q'_{j,1} \cup Q'_{j,2} \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\}\big\rangle$ for all $0 \le j < \hat{\imath}$, and if $\hat{\imath} < \omega$, let $L\big(\langle v_{\hat{\imath}}, V_{\hat{\imath}+1}\rangle\big) \stackrel{def}{=} \langle q_I, (\theta_{\hat{\imath},1} \wedge \theta_{\hat{\imath},2}), \emptyset, Q'_{\hat{\imath},1} \cup Q'_{\hat{\imath},2}\rangle$.

We check that G is a partial run of $\mathcal{A}$ on w. By the definition of G, $V_0 = \{v_0\}$, $L(v_0) = q_I$, V consists of finite disjoint levels, and E is a collection of edges between consecutive levels of G.

Let $v \in V_j$ for some $0 \le j < \omega$. Then v either has no outgoing edges, or $v = v_j$, $L(v) = q_I$, and v has the unique outgoing edge $\langle v_j, V_{j+1}\rangle \in E$. By definition of the edge sequences, $w(j) \models (\theta_{j,1} \wedge \theta_{j,2})$. 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 $t = \big\langle q_I, (\theta_{j,1} \wedge \theta_{j,2}), F, Q'\big\rangle$, where either $F = F_{j,1} \cup F_{j,2}$ and $Q' = (Q'_{j,1} \cup Q'_{j,2} \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\}$, or $F = \emptyset$ and $Q' = Q'_{j,1} \cup Q'_{j,2}$ (by the choice of $\hat{\imath}$, the latter case occurs iff $\hat{\imath} < \omega$ and $j = \hat{\imath}$). From the definitions it now follows that $L\big(\langle v_j, V_{j+1}\rangle\big) = t$. Likewise, it is straightforward to check that $L(V_{j+1}) = Q'$, and thus the edge labelling is consistent.

By the definition of G, each node $v_j \in V$ for some $1 \le j < \omega$ is a successor of the node $v_{j-1} \in V$.

We show that G is a partial fin-accepting run of $\mathcal{A}$ on w. This holds trivially if $\hat{\imath} < \omega$, in particular, if the edge sequences $(t_{j,k})_{0 \leq j < i+1}$ satisfy condition (a) of the above claim. Otherwise $i = \hat{\imath} = \omega$, and $\mathcal{E}(G)$ contains the unique acceptance sequence $r = \big(\langle v_j, V_{j+1}\rangle\big)_{0 \leq j < \omega}$. Assume that $\mathrm{fin}(r) \neq \emptyset$. Then there exists an index $0 \leq l < \omega$ and an acceptance condition $f \in \mathcal{F}$ shared by every transition $L\big(\langle v_j, V_{j+1}\rangle\big) = \big\langle q_I, (\theta_{j,1} \wedge \theta_{j,2}), F_{j,1} \cup F_{j,2}, (Q'_{j,1} \cup Q'_{j,2} \cup \{q_I\}) \setminus \{q_{I1}, q_{I2}\}\big\rangle$ for all $l \leq j < \omega$.

We claim that there now exists a $k \in \{1, 2\}$ and an index $l \leq \hat{l} < \omega$ such that $(t_{j,k})_{\hat{l} \leq j < \omega}$ is an infinite sequence of labels of consecutive edges (i.e., an edge sequence r') embedded in G^1 or G^2 in which $f \in F_{j,k}$ actually holds for all $\hat{l} \leq j < \omega$. Because $\mathrm{fin}(r') \neq \emptyset$, this then implies that one of the runs G^1 or G^2 cannot be fin-accepting, which contradicts our original assumptions. Therefore it must be the case that $\mathrm{fin}(r) = \emptyset$, and G is a partial fin-accepting run of $\mathcal{A}$ on w.

In case (b), there exists an index $0 \leq i' < \omega$ such that $t_{j,k}$ is an initial self-loop of $\mathcal{A}_k$ for all $i' \leq j < \omega$ and $k \in \{1, 2\}$. Because the edges of r are self-loops of $\mathcal{A}$ for all $0 \leq j < \omega$, it follows from the definition of $\mathcal{A}$ that $F_{j,1} \cap \mathcal{F}_1 \cap \mathcal{F}_2 = F_{j,2} \cap \mathcal{F}_1 \cap \mathcal{F}_2$ holds for all $\hat{l} \stackrel{def}{=} \max\{l, i'\} \leq j < \omega$.

The above claim is now easily seen to hold if $f \in F_{j,1} \cap F_{j,2}$ holds for all $\hat{l} \leq j < \omega$. Otherwise there exists an index $\hat{l} \leq l' < \omega$ such that $f \in F_{l',k} \setminus F_{l',3-k}$ for some $k \in \{1, 2\}$. But then $f \notin \mathcal{F}_1 \cap \mathcal{F}_2$, and thus $f \in F_{j,3-k}$ cannot hold for any $0 \leq j < \omega$. Therefore it must be the case that $f \in F_{j,k}$ holds for all $\hat{l} \leq j < \omega$.

In case (c), the transition sequence $(t_{j,k})_{0 \leq j < \omega}$ is an infinite sequence of initial self-loops of $\mathcal{A}_k$ for some $k \in \{1, 2\}$, and there exist infinitely many indices $0 \leq j < \omega$ such that $t_{j,3-k}$ is an initial non-self-loop transition of $\mathcal{A}_{3-k}$. Because both $\mathcal{A}_1$ and $\mathcal{A}_2$ are simplified in the sense of Corollary 2.3.17, $F_{j,3-k} = \emptyset$ for each such j, and thus the fact that $f \in F_{j,1} \cup F_{j,2}$ holds for all $l \leq j < \omega$ implies that $f \in F_{j,k} \subseteq \mathcal{F}_k$. Suppose that there exists an index $l \leq l' < \omega$ such that $f \notin F_{l',k}$. Then $f \in F_{l',3-k} \subseteq \mathcal{F}_{3-k}$, which implies that $f \in \mathcal{F}_1 \cap \mathcal{F}_2$, and thus (again, due to the automaton simplification) $t_{l',3-k}$ is an initial self-loop of $\mathcal{A}_{3-k}$. But then the fact that $F_1 \cap \mathcal{F}_1 \cap \mathcal{F}_2 = F_2 \cap \mathcal{F}_1 \cap \mathcal{F}_2$ holds by the definition of $\mathcal{A}$ implies that $f \in F_{l',k}$, a contradiction. Therefore $f \in F_{j,k}$ holds for all $\hat{l} \stackrel{def}{=} l \leq j < \omega$, and the claim follows.

Finally, if $v \in V_j$ is a node with no outgoing edges in G, then $1 \leq j < \omega$, $v_{j-1,k,l}$ for some $k \in \{1, 2\}$ and $1 \leq l \leq n_{j-1,k}$, and v is labelled with a node in $Q'_{j-1,k}$. Because $\mathcal{A}_k^q$ has a fin-accepting run on w^j for all $q \in Q'_{j-1,k}$ (the transition $t_{j-1,k}$ always labels an edge starting from a node at level $j - 1$ in one of the fin-accepting runs G^1 and G^2), it follows that G can be extended into a full fin-accepting run of $\mathcal{A}$ on w (Proposition 2.3.10), and therefore $w \in \mathcal{L}_{\mathrm{fin}}(\mathcal{A})$. $\square$

By using the above result in the proof of Theorem 3.4.3, it follows that the formula translation remains correct if the set of translation rules is extended with the one given in the proposition. The effects of this translation rule on the translation will be discussed in Sect. 4.3.4.

4.3.3 Modified Translation Rules: Binary Temporal Connectives

In this section we introduce new translation rules for the Until connectives and use them to derive new rules also for the Release connectives. Similar to the previous section, the new rules apply to the basic translation of an Until formula $(\varphi_1 \mathsf{U} \varphi_2)$ only in certain special cases, more specifically, under the language containment relationship $\mathcal{L}(\varphi_2) \subseteq \mathcal{L}(\varphi_1)$ (which can be checked for using the basic translation as described in Sect. 4.2). Even though the new rules for the Until connectives depend on language containment relationships, we still obtain improved translation rules independent of any language containment relationships for the Release connectives.

The basic translation rule for an Until type formula $(\varphi_1 \mathsf{U} \varphi_2)$ creates a new initial state for the constructed automaton and then transforms the initial transitions of the automaton built for the subformula φ_1 into initial self-loops of the compound automaton. Clearly, all initial self-loops of this component automaton are unrolled in the application of the rule, and thus the initial state q of the component automaton will remain reachable from the initial state of any automaton built from the compound automaton in subsequent translation steps. However, if $\mathcal{L}(\varphi_2) \subseteq \mathcal{L}(\varphi_1)$, it proves possible to defer the introduction of a self-loop having q in its target states in the compound automaton by a slight modification to the way in which the initial self-loops of this automaton are defined. This increases the opportunities for removing q later from the final automaton.

The following proposition gives a formal definition for the new translation rules and shows, analogous to Lemma 3.4.2, that the acceptance behavior of a linear alternating automaton built using one of these rules still corresponds to the semantics of the Until connectives under the given assumption on language containment. As usual, we replace language containment between LTL formulas with language containment between linear alternating automata.

Proposition 4.3.3 *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 linear alternating automata such that $\mathcal{A}_1^q = \mathcal{A}_2^q$ holds for all $q \in Q_1 \cap Q_2$, and assume that $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_1)$. Define the linear 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 $q_I \notin Q_1 \cup Q_2$), $\Delta \stackrel{def}{=} \Delta_1 \cup \Delta_2 \cup \Delta'$, and $\mathcal{F} \stackrel{def}{=} \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}'$, where either*

$$\mathcal{F}' \stackrel{def}{=} \{f\}, \quad \Delta' \stackrel{def}{=} \big\{\langle q_I, \theta, \{f\}, (Q' \setminus \{q_{I1}\}) \cup \{q_I\}\rangle \,\big|\, \langle q_{I1}, \theta, F, Q'\rangle \in \Delta_1\big\} \cup \big\{\langle q_I, \theta, \emptyset, Q'\rangle \,\big|\, \langle q_{I2}, \theta, F, Q'\rangle \in \Delta_2\big\}$$

$(f \notin \mathcal{F}_1 \cup \mathcal{F}_2)$, or

$$\mathcal{F}' \stackrel{def}{=} \emptyset, \quad \Delta' \stackrel{def}{=} \big\{\langle q_I, \theta, F, (Q' \setminus \{q_{I1}\}) \cup \{q_I\}\rangle \,\big|\, \langle q_{I1}, \theta, F, Q'\rangle \in \Delta_1\big\} \cup \big\{\langle q_I, \theta, \emptyset, Q'\rangle \,\big|\, \langle q_{I2}, \theta, F, Q'\rangle \in \Delta_2\big\}.$$

Then, 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 $\mathcal{A}_1$ fin-accepts w^j for all $0 \le j < i$, or $\mathcal{F}' = \emptyset$ and $\mathcal{A}_1$ fin-accepts w^i for all $0 \le i < \omega$.

Proof: ("$\Rightarrow$") Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$. Thus $\mathcal{A}$ has a fin-accepting run $G = \langle V, E, L\rangle$ on w, and there exists (Proposition 2.3.7) an index $0 \le i \le \omega$ and a sequence of consecutive edges $(e_j)_{0\le j<i+1}$, $e_j \in E \cap (V_j \times 2^{V_{j+1}})$, such that $L(e_j)$ is an initial self-loop of $\mathcal{A}$ for all $0 \le j < i$, and if $i < \omega$, then $L(e_i)$ is an initial transition of $\mathcal{A}$ that is not a self-loop.

We first show that if $\mathcal{A}_1$ fin-accepts w^j for some $1 \le j < i + 1$, then $\mathcal{A}_1$ fin-accepts w^k for all $0 \le k \le j$. Assume that $\mathcal{A}_1$ fin-accepts w^j for some $1 \le j < i+1$. Because $j - 1 < i$, $L(e_{j-1}) = \langle q_I, \theta, F, Q'\rangle \in \Delta$ is an initial self-loop of $\mathcal{A}$, and because G is a run, $w(j-1) \models \theta$, and Q' is the union of the labels of the target nodes of e_{j-1} in G. The fact that G is fin-accepting now implies that $\mathcal{A}^{q'}$ fin-accepts w^j for all $q' \in Q'$ by Proposition 2.3.9. Because $L(e_{j-1})$ is an initial self-loop of $\mathcal{A}$, there exists a transition $\langle q_{I1}, \theta, F', Q''\rangle \in \Delta_1$ for some $F' \subseteq \mathcal{F}_1$ and $Q'' \subseteq Q_1$ such that $Q' = \big(Q'' \setminus \{q_{I1}\}\big) \cup \{q_I\}$ by the definition of $\mathcal{A}$. It follows that $\mathcal{A}^{q'}$ $(= \mathcal{A}_1^{q'})$ fin-accepts w^j for all $q' \in Q'' \setminus \{q_{I1}\}$.

If $q_{I1} \notin Q''$, then $\mathcal{A}_1^{q'}$ fin-accepts w^j for all $q' \in Q''$. Otherwise $Q'' = \big(Q' \setminus \{q_I\}\big) \cup \{q_{I1}\}$, and $\mathcal{A}^{q'}$ $(= \mathcal{A}_1^{q'})$ fin-accepts w^j for all $q' \in Q' \setminus \{q_I\}$. By the induction hypothesis, it follows that also $\mathcal{A}_1^{q_{I1}}$ necessarily fin-accepts w^j, and thus $\mathcal{A}_1^{q'}$ fin-accepts w^j for all $q' \in Q''$ also in this case.

Because $w(j-1) \models \theta$, we can now apply Lemma 3.4.1 to conclude that $\mathcal{A}_1$ fin-accepts w^{j-1}. By induction, it follows that $\mathcal{A}_1$ fin-accepts w^k for all $0 \le k \le j$.

If $i < \omega$, then $L(e_i) = \langle q_I, \theta, F, Q'\rangle$ is not an initial self-loop of $\mathcal{A}$. By the definition of $\mathcal{A}$, $L(e_i)$ corresponds to a transition $\langle q_{I2}, \theta, F', Q'\rangle \in \Delta_2$ for some $F' \subseteq \mathcal{F}_2$. Again, because G is a fin-accepting run of $\mathcal{A}$ and $w(i) \models \theta$, $\mathcal{A}^{q'}$ $(= \mathcal{A}_2^{q'})$ fin-accepts w^{i+1} for all $q' \in Q'$ (Proposition 2.3.9), and by Lemma 3.4.1, it follows that $\mathcal{A}_2$ fin-accepts w^i. Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$, we conclude that also $\mathcal{A}_1$ fin-accepts w^i. By the above inductive argument, $\mathcal{A}_1$ fin-accepts w^j for all $0 \le j \le i$, and the result follows.

On the other hand, if $i = \omega$, then $L(e_j)$ is an initial self-loop of $\mathcal{A}$ for all $0 \le j < \omega$. Then necessarily $\mathcal{F}' = \emptyset$, since otherwise $\text{fin}\big((e_j)_{0\le j<\omega}\big) \ne \emptyset$, which would contradict the assumption that G is fin-accepting. Because each initial self-loop of $\mathcal{A}$ corresponds to an initial transition of $\mathcal{A}_1$ by the definition of $\mathcal{A}$, then the result follows immediately if infinitely many of these transitions are *not* initial self-loops of $\mathcal{A}_1$ by the above inductive argument (if $L(e_j)$ corresponds to a transition of $\mathcal{A}_1$ that is not a self-loop of $\mathcal{A}_1$, then j can be used as the base case for the induction).

Otherwise there exists an index $0 \le j < \omega$ 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 $0 \le k < \omega$. Let $e_j = \langle \hat{v}_0, V'\rangle$ for some $\hat{v}_0 \in V_j$ and $V' \subseteq V_{j+1}$, and let $L(e_{j+k}) = \langle q_I, \theta'_k, F'_k, Q'_k\rangle$, where $w(j+k) \models \theta'_k$ and $Q'_k \setminus \{q_I\} = \{q_{k,1}, q_{k,2}, \ldots, q_{k,n_k}\}$ $(0 \le n_k < \omega)$ for each $0 \le k < \omega$.

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 $0 \le k < \omega$,
- $E' \overset{\text{def}}{=} \bigcup_{0\le k<\omega} \big\{\langle \hat{v}_k, V'_{k+1}\rangle\big\}$,
- $L'(\hat{v}_k) \overset{\text{def}}{=} q_{I1}$, $L'(v_{k,l}) \overset{\text{def}}{=} q_{k,l}$ for all $0 \le k < \omega$ and $1 \le l \le n_k$, and

$$L'\big(\langle \hat{v}_k, V'_{k+1}\rangle\big) \overset{def}{=} \big\langle q_{I1}, \theta'_k, F'_k, (Q'_k \setminus \{q_I\}) \cup \{q_{I1}\}\big\rangle \text{ for all } 0 \leq k < \omega.$$

G' is a partial run of $\mathcal{A}_1$ on w^j:

- $V'_0 = \{\hat{v}_0\}$, $L'(\hat{v}_0) = q_{I1}$, and V' is partitioned into disjoint finite levels (with edges only between successive levels).
- Let $v \in V'_k$ for some $0 \leq k < \omega$. Then v either has no outgoing edges and satisfies the second condition of a partial run trivially, or $v = \hat{v}_k$. In this case v has the unique outgoing edge $e = \langle v, V'_{k+1}\rangle \in E'$. Because $L(e_{j+k}) = \langle q_I, \theta'_k, F'_k, Q'_k\rangle$ is a self-loop of $\mathcal{A}$ that corresponds to an initial self-loop of $\mathcal{A}_1$ and $\mathcal{F}' = \emptyset$, it follows from the definition of $\mathcal{A}$ that Δ_1 contains the transition $\big\langle q_{I1}, \theta'_k, F'_k, (Q'_k \setminus \{q_I\}) \cup \{q_{I1}\}\big\rangle = L'(e) = \big\langle L'(v), \theta'_k, F'_k, L(V'_{k+1})\big\rangle$. Furthermore, because G is a run of $\mathcal{A}$, $w(j+k) = w^j(k) \models \theta'_k$, and thus the edge labelling is consistent.
- By the definition of E', each node $v \in V'_k$ for some $1 \leq k < \omega$ is a successor of a node at level $k-1$ of G', and thus G' satisfies also the third condition of a partial run of $\mathcal{A}_1$ on w^j.

Clearly, $\mathcal{E}(G')$ contains the unique acceptance sequence $r = (e'_k)_{0\leq k<\omega} = \big(\langle \hat{v}_k, V'_{k+1}\rangle\big)_{0\leq k<\omega}$. Because the transition $L'(e'_k)$ inherits its acceptance conditions from the transition $L(e_{j+k})$ for all $0 \leq k < \omega$, it follows that $\mathrm{fin}(r) = \mathrm{fin}\big((e'_k)_{0\leq k<\omega}\big) = \mathrm{fin}\big((e_{j+k})_{0\leq k<\omega}\big) = \emptyset$, because G is a fin-accepting run of $\mathcal{A}$ on w. Thus G' is a partial fin-accepting run of $\mathcal{A}_1$ on w^j.

If $v \in V'_k$ for some $0 \leq k < \omega$ has no outgoing edges, then necessarily $v = v_{k-1,l}$ for some $1 \leq k < \omega$ and $1 \leq l \leq n_{k-1}$, and $L'(v) = q_{k-1,l} \in Q'_{k-1} \setminus \{q_I\} \subseteq Q_1$. Because G is a fin-accepting run of $\mathcal{A}$ on w and $e_{j+k-1} \in E$ and $L(e_{j+k-1}) = \langle q_I, \theta'_{k-1}, F'_{k-1}, Q'_{k-1}\rangle$, it follows that $w(j+k-1) \models \theta'_{k-1}$, and $\mathcal{A}^{q'}$ fin-accepts $w^{j+k} = (w^j)^k$ for all $q' \in Q'_{k-1}$. Thus especially $\mathcal{A}^{L'(v)}$ $(= \mathcal{A}_1^{L'(v)})$ fin-accepts w^j, and because v is arbitrary, G' can be extended into a fin-accepting run of $\mathcal{A}_1$ on w^j by Proposition 2.3.10.

Since each level of the fin-accepting run G' includes a node labelled with q_{I1}, it follows by Proposition 2.3.9 that $\mathcal{A}_1$ fin-accepts $(w^j)^k = w^{j+k}$ for all $0 \leq k < \omega$. By the inductive argument given in the beginning of the proof, $\mathcal{A}_1$ fin-accepts w^k also for all $0 \leq k \leq j$, and thus $\mathcal{A}_1$ fin-accepts w^k for all $0 \leq k < \omega$.

("$\Leftarrow$") This result follows from an obvious modification of the proof of the corresponding direction in Lemma 3.4.2. □

The above result can be used directly in the proof of Theorem 3.4.3 to show that the automaton translation remains correct by replacing the original translation rules for the $\mathsf{U_s}$ and the $\mathsf{U_w}$ connectives (Table 3.1, page 33) with the ones shown in the upper half of Table 4.2. See also Figs. 4.7 (b) and (c).

The rules for the $\mathsf{R_s}$ and $\mathsf{R_w}$ 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

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

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 any two LTL formulas $\varphi_1, \varphi_2 \in LTL(AP)$, it follows that the language

Table 4.2: Refined translation rules for the binary temporal connectives

$\circ$	$\mathcal{F}_\circ$	$\Delta_\circ$
$\mathsf{U_s}$	$\{f\}$	$\left\{\langle q_I, \theta_1, \{f\}, (Q'_1 \setminus \{q_{I1}\}) \cup \{q_I\}\rangle \mid \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\right\} \cup \left\{\langle q_I, \theta_2, \emptyset, Q'_2\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\}$ $\boxed{\textit{if } \mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)}$
		$\left\{\langle q_I, \theta_1, \{f\}, Q'_1 \cup \{q_I\}\rangle \mid \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\right\} \cup \left\{\langle q_I, \theta_2, \emptyset, Q'_2\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\}$ $\boxed{\textit{otherwise}}$
$\mathsf{U_w}$	$\emptyset$	$\left\{\langle q_I, \theta_1, F_1, (Q'_1 \setminus \{q_{I1}\}) \cup \{q_I\}\rangle \mid \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\right\} \cup \left\{\langle q_I, \theta_2, \emptyset, Q'_2\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\}$ $\boxed{\textit{if } \mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)}$
		$\left\{\langle q_I, \theta_1, \emptyset, Q'_1 \cup \{q_I\}\rangle \mid \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\right\} \cup \left\{\langle q_I, \theta_2, \emptyset, Q'_2\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\}$ $\boxed{\textit{otherwise}}$
$\mathsf{R_s}$	$\{f\}$	$\left\{\langle q_I, \theta_2, \{f\}, (Q'_2 \setminus \{q_{I2}\}) \cup \{q_I\}\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\} \cup \left\{\langle q_I, \theta_1, \emptyset, Q'_1\rangle \mid \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\right\}$ $\boxed{\textit{if } \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_2)}$
		$\left\{\langle q_I, \theta_2, \{f\}, (Q'_2 \setminus \{q_{I2}\}) \cup \{q_I\}\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\} \cup \left\{\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q'_1 \cup Q'_2\rangle \mid \begin{array}{l}\langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1,\\ \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\end{array}\right\}$ $\boxed{\textit{otherw.}}$
$\mathsf{R_w}$	$\emptyset$	$\left\{\langle q_I, \theta_2, F_2, (Q'_2 \setminus \{q_{I2}\}) \cup \{q_I\}\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\} \cup \left\{\langle q_I, \theta_1, \emptyset, Q'_1\rangle \mid \langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1\right\}$ $\boxed{\textit{if } \mathcal{L}_{\text{fin}}(\mathcal{A}_1) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_2)}$
		$\left\{\langle q_I, \theta_2, F_2, (Q'_2 \setminus \{q_{I2}\}) \cup \{q_I\}\rangle \mid \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\right\} \cup \left\{\langle q_I, (\theta_1 \wedge \theta_2), \emptyset, Q'_1 \cup Q'_2\rangle \mid \begin{array}{l}\langle q_{I1}, \theta_1, F_1, Q'_1\rangle \in \Delta_1,\\ \langle q_{I2}, \theta_2, F_2, Q'_2\rangle \in \Delta_2\end{array}\right\}$ $\boxed{\textit{otherw.}}$

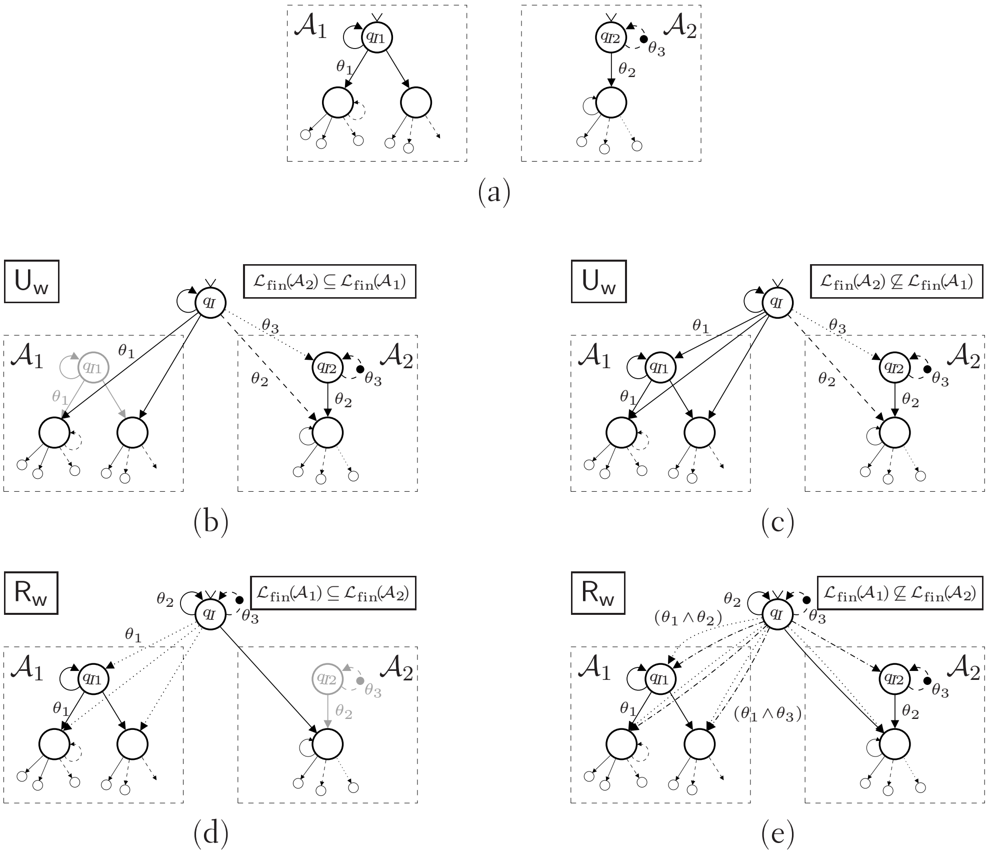


Fig. 4.7: Automata built using the modified translation rules for the weak binary temporal connectives. (a) Two component automata $\mathcal{A}_1$ and $\mathcal{A}_2$; (b) Automaton built from $\mathcal{A}_1$ and $\mathcal{A}_2$ with the $\mathsf{U_w}$ translation rule under the assumption $\mathcal{L}_{\text{fin}}(\mathcal{A}_2) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_1)$; (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 $\mathcal{L}_{\text{fin}}(\mathcal{A}_1) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_2)$; (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

containment assumption in Proposition 4.3.3 holds trivially between the top-level subformulas of the Until formulas corresponding to the Release formulas. This 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 4.1 allow a further improvement in the translation of Release formulas of the form $(\varphi_1 \mathsf{R}\, \varphi_2)$, this time under the 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 $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$ implies the language containment assumption in Proposition 4.3.3 for the Until formulas, the translation of the Release formulas reduces to the translation of Until formulas using the improved translation rules. (As noted in Sect. 4.3.1, we may not need to apply a rule explicitly if we are able to reuse an automaton built for an Until formula.) Using the Until 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 4.2; see also Figs. 4.7 (d) and (e) for illustration.

We end this section with an example that provides some justification for the necessity of the language containment assumption in the application of the new rules to formulas with $\mathsf{U_s}$ or $\mathsf{U_w}$ as their main connective.

Example 4.3.4 Figure 4.8 shows two automata built for the LTL formula

$$\big((\mathsf{G}p_1)\,\mathsf{U_s}\,p_2\big) \equiv \big((\bot\,\mathsf{R_w}\,p_1)\,\mathsf{U_s}\,p_2\big) \in LTL\big(\{p_1, p_2\}\big),$$

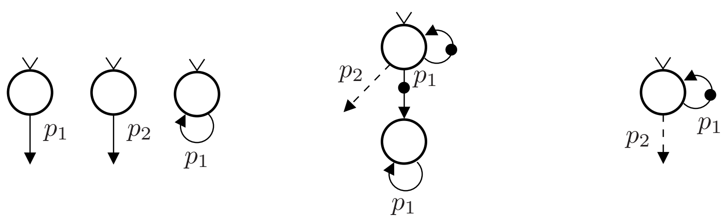
where the automaton (b) is obtained using the original translation rules (with the usual transition guard simplification and restriction of the automaton to states reachable from its initial state), while the automaton (c) is (erroneously) obtained with the new rules by ignoring the requirement concerning the relationship between the languages of the automata built for the subformulas $(\bot\,\mathsf{R_w}\,p_1)$ and p_2; clearly, $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}_{p_2})$ $(= \mathcal{L}(p_2)) \subseteq \mathcal{L}_{\mathrm{fin}}(\mathcal{A}_{(\bot\,\mathsf{R_w}\,p_1)})$ $(= \mathcal{L}(\mathsf{G}p_1))$ does not hold in this case. It is easy to see that the automaton (c) fin-accepts the word $\{p_1\}\{p_2\}^\omega$, which is, however, not a model of the LTL formula $\big((\mathsf{G}p_1)\,\mathsf{U_s}\,p_2\big)$. ■

4.3.4 Discussion

In this section we illustrate and discuss some effects of substituting the modified translation rules for the original rules given in Sect. 3.2 in the translation procedure.

Translation Example Revisited

We first illustrate the application of the refined translation rules by repeating the translation for the formula from which we built a linear alternating automaton in Sect. 4.1.3 using the basic rules.



(a) (b) (c)

Fig. 4.8: The improved Until translation rules cannot always be 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 improved translation rules, ignoring the requirement on the language containment relationship between the languages fin-accepted by these automata

Example 4.3.5 Consider again the LTL formula

$$\begin{aligned}
&\Big(\big(\mathsf{GF}p_1 \wedge \mathsf{GF}p_2\big) \vee \big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)\Big)\\
\equiv\ &\Big[\Big(\big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_1)\big) \wedge \big(\bot\,\mathsf{R_w}\,(\top\,\mathsf{U_s}\,p_2)\big)\Big) \vee \big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)\Big]\\
\equiv\ &(\varphi \vee \psi);
\end{aligned}$$

similar to the example in Sect. 4.1.3, we translate the formula into an automaton by dealing with its top-level subformulas φ and ψ separately.

Because we do not have new translation rules for atomic formulas, we obtain the same automata as in Sect. 4.1.3 for the atomic formulas of φ (repeated in Fig. 4.9 (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. 4.9 (b). (Actually, because $\mathcal{A}_{\top}$ has no initial self-loops, we obtain in this case the same compound automata as previously.)

Because $\mathcal{L}_{\text{fin}}(\bot) = \emptyset$, the language fin-accepted by $\mathcal{A}_{\bot}$ is trivially a subset of the language fin-recognized by any automaton. Therefore, by using the appropriate $\mathsf{R_w}$ rule that makes use of this language containment assumption, we obtain the automata shown in Fig. 4.9 (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 again remove the transitions with unsatisfiable guards from these automata (Fig. 4.9 (d)). Because these automata do not share common acceptance conditions, Proposition 4.3.2 allows us to build an automaton for the formula φ by merging all pairs of initial self-loops of these automata. This step results in the single-state automaton shown in Fig. 4.9 (e). Figure 4.9 (f) shows the same automaton after transition guard simplification. Using the refined translation rules, we are thus able to replace the five-state automaton in Fig. 4.1 (f) (page 52) with an automaton consisting of only a single state.

We then repeat the translation for the formula ψ. As before, we start from the automata for the atomic formulas (Fig. 4.10 (a)). Because $\mathcal{L}_{\text{fin}}(\mathcal{A}_{p_4}) \nsubseteq \mathcal{L}_{\text{fin}}(\mathcal{A}_{p_5})$, we apply the general refined $\mathsf{R_s}$ rule to define an automaton for the formula $(p_4\,\mathsf{R_s}\,p_5)$ (Fig. 4.10 (b)). The weak version of the same rule needs to be applied to build an automaton for the formula $\big(p_3\,\mathsf{R_w}\,(p_4\,\mathsf{R_s}\,p_5)\big)$

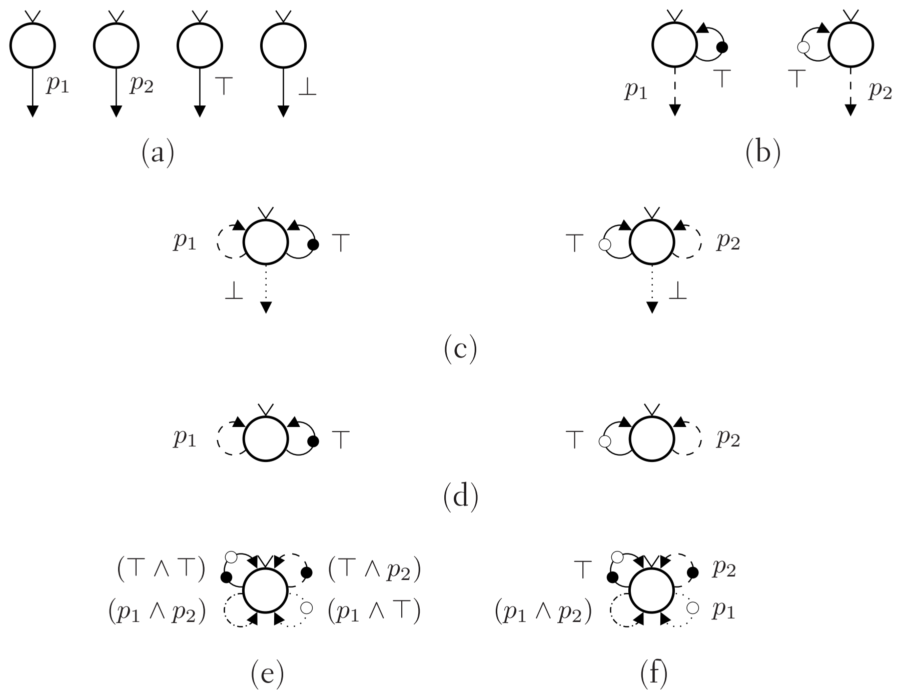


Fig. 4.9: Building an automaton for the LTL formula $\varphi \overset{def}{\equiv} \Big(\big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)\big) \wedge \big(\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)\big)\Big)$ using the refined translation rules. (a) Automata for the atomic subformulas of 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) The automata obtained from (c) by removing transitions with unsatisfiable guards; (e) Automaton for the formula φ; (f) The automaton obtained from (e) by transition guard simplification

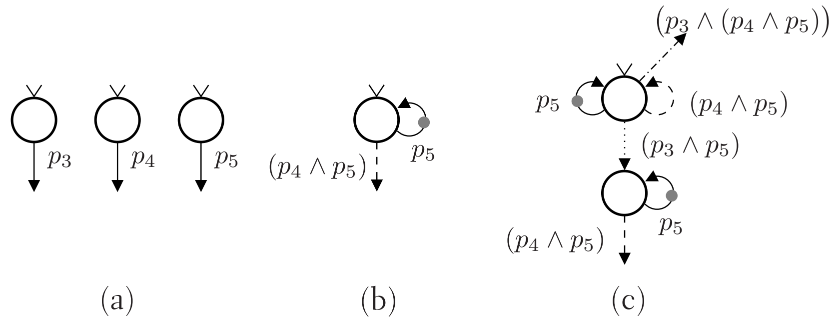


Fig. 4.10: Building an automaton for the formula $\psi \stackrel{def}{\equiv} \big(p_3 \, \mathsf{R_w} \, (p_4 \, \mathsf{R_s} \, p_5)\big)$ using the refined translation rules. (a) Automata for the atomic subformulas of ψ; (b) Automaton for the formula $(p_4 \, \mathsf{R_s} \, p_5)$; (c) Automaton for the formula ψ

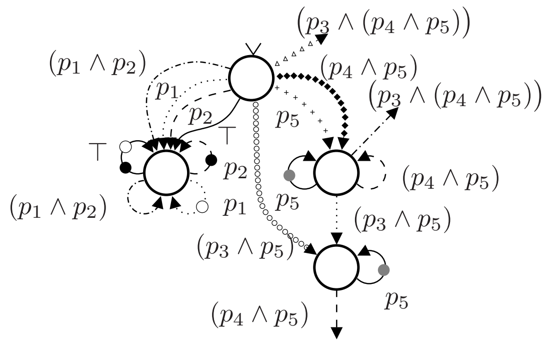


Fig. 4.11: Automaton constructed for the LTL formula $\Big[\Big(\big(\bot \, \mathsf{R_w} \, (\top \, \mathsf{U_s} \, p_1)\big) \wedge \big(\bot \, \mathsf{R_w} \, (\top \, \mathsf{U_s} \, p_2)\big)\Big) \vee \big(p_3 \, \mathsf{R_w} \, (p_4 \, \mathsf{R_s} \, p_5)\big)\Big]$ using the refined translation rules

(Fig. 4.10 (c)). In comparison to the automaton obtained in Sect. 4.1.3 (Fig. 4.2 (c), page 53), 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

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

by applying the $\vee$ rule as shown in Fig. 4.11. This automaton has three states less than the automaton we previously obtained for the same formula in Sect. 4.1.3 (cf. Fig. 4.3, page 53). Additionally, no transition of 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. We shall investigate such automata more closely in Sect. 5.1. ■

Similarities and Differences between the Modified and the Original Rules The modified translation rules behave very much like the original rules given in Sect. 3.2. For example, each new translation rule adds a new initial state to the automaton (preserving the correspondence between states in the automaton and subformulas of a given formula φ), and because none of the new rules changes the transition structure of the component automata to which the rule is applied, the compound automata will again be linear alternating automata. Also all transition guards will remain conjunctions of atomic formulas.

The first difference between the original and the modified rules concerns the handling of acceptance conditions. Recall from the discussion in Sect. 3.2 that every transition that has a nonempty set of acceptance conditions in an automaton built using the original rules always starts from a state corresponding to a strong temporal eventuality. 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 modified rules, however, change this behavior: although the rules for the strong temporal eventualities 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 starting from one of its own target states. Intuitively, the acceptance conditions may thus propagate "upward" in an automaton during its incremental construction via chains of states corresponding to nested weak eventualities or conjunctions in the given LTL formula, where each chain starts from a state corresponding to a strong temporal eventuality subformula. Therefore, an automaton built using the modified rules may contain self-loops that are associated with the same acceptance condition in spite of their different start states; additionally, as illustrated by the previous example, applications of the modified $\wedge$ rule may create states with self-loops associated with multiple acceptance conditions.

Clearly, changing the translation rules also forces us to reconsider the upper bound obtained in Sect. 3.3 for the size of a linear alternating automaton corresponding to a given LTL formula φ. Recall that the contribution of any automaton built for some subformula $\psi \in \mathrm{Sub}(\varphi)$ to the size of 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 modified 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 requires that both component automata have initial self-loops.) Additionally, provided that neither component automaton is a trivial automaton with no initial transitions, the initial state of the compound automaton will always be reachable from itself. It is easy to see that this consideration applies also to the modified translation rules (for example, the rules for the Until connectives with a negative language containment assumption are identical to the original rules; other rules may only allow a reduction in the number of target states of some initial transitions of the compound automaton). Therefore, the number of states reachable from the initial state 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 3.3.2 is still valid if only the new rules for the binary temporal connectives are considered.

Unfortunately, the reasoning used in Sect. 3.3 does not apply to the modified translation rule for the $\wedge$ connective. Due to the possible introduction of initial self-loops to the compound automaton when applying this rule, the initial state of the compound automaton may become reachable from itself in the automaton; obviously, this can never occur when using the original translation rules. Because of these initial self-loops, the initial state of the compound automaton will thus remain reachable also from any automaton obtained from this automaton using further translation rules. This fact clearly breaks the upper bound given in Corollary 3.3.2 for the size of the final automaton for a given LTL formula φ; 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 performs better than the new rule.

Example 4.3.6 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 modified 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. 4.12 (a). On the other hand, using the modified translation rule to build an automaton for the same formula will cause the self-loops starting from the initial states of the component automata to be merged into an initial self-loop of the compound automaton (Fig. 4.12 (b)).

Applying the $\vee$ rule to the automaton built using the original rules results in an automaton, where the initial state of the automaton shown in Fig. 4.12 (a) is not reachable from the initial state of the compound automaton (Fig. 4.12 (c)). However, applying the same rule to the automaton in Fig. 4.12 (b) will force the initial self-loop of this automaton to be “unrolled” by the rule. It follows that no states can be removed from the resulting automaton, and thus the result is suboptimal in comparison to the automaton in Fig. 4.12 (c). ■

From the above example, we see that the assumption that two branches of *any* accepting run of a compound automaton built using the original $\wedge$ rule can always be merged such that the run does not have to visit the initial state of one of the component automata is obviously very optimistic and should therefore be used only sparingly. Because the modified translation rule for the $\wedge$ connective nevertheless has also clear advantages over the original rule in some cases (cf. Example 4.3.5), a heuristic decision is needed on when the rule should be applied. For example, a very simple strategy that prevents the behavior shown in Example 4.3.6 is to always first use the new rule for translation and then revert back to using the original rule if the constructed compound automaton has initial transitions (with nonempty guards) to the initial states of its component automata.

4.4 REMOVING REDUNDANT TRANSITIONS

The simplification heuristics obtained by modifying the original translation rules are essentially based on replacing transitions of linear alternating au-

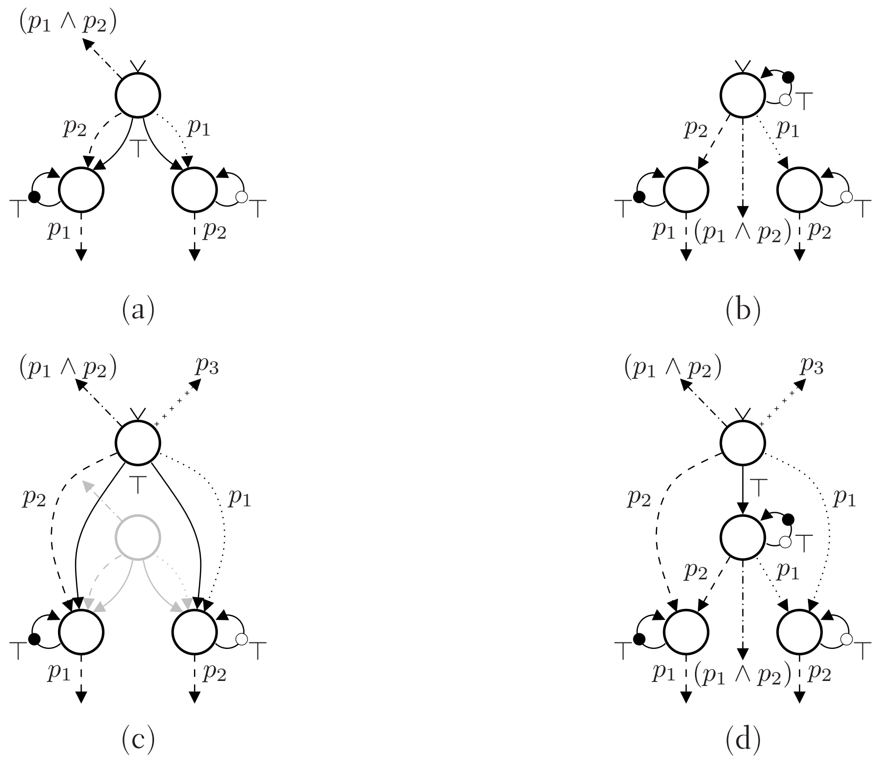


Fig. 4.12: Greedy application of the refined translation rule for the $\wedge$ connective may lead to an increase in the size of the generated automata. (a)–(b) Automata built for the formula $(\mathsf{F}p_1 \wedge \mathsf{F}p_2)$ using the original (a) and refined rules (b), respectively; (c)–(d) Automata 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

tomata with transitions having fewer target states, in a way that preserves the language of the automaton. However, these heuristics can guarantee a reduction in the actual number of transitions or states in the automaton only in a few special cases. In this section, we shall concentrate on the more specific problem of finding *redundant* transitions that can be removed from a linear alternating automaton without changing its language. Formally, given an automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$, a transition $t \in \Delta$ is *redundant* iff $\mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}_{\text{fin}}(\mathcal{A}')$ holds for 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 linear alternating automaton, then so is $\mathcal{A}'$, because the number of loops in $\mathcal{A}$ cannot increase in the modification. Therefore all simplification heuristics specific to linear alternating automata remain applicable to $\mathcal{A}'$ and any automaton obtained from it by removing more transitions.)

The incremental strategy of building increasingly complex automata from simpler automata (as used in the formula translation) can easily be combined with on-the-fly transition redundancy analysis. Because no automaton $\mathcal{A}$ built from some subformula ψ of an LTL formula φ using some translation rule will have its structure altered by any translation rule that uses $\mathcal{A}$ as a component, the language of $\mathcal{A}$ remains fixed regardless of the way the translation of φ proceeds after $\mathcal{A}$ has been constructed. This implies that $\mathcal{A}$ can be checked for redundant transitions immediately when its definition is complete. In particular, removing all redundant initial transitions of $\mathcal{A}$ as soon as possible may reduce 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. Similar on-the-fly transition redundancy analysis is conceptually more difficult to combine with direct tableau-based translations from LTL to nondeterministic automata, which in their basic form necessitate the full construction of an automaton for φ before full-scale state or transition redundancy analysis is possible. The requirement for access to a full automaton for φ is implicit also in the design of many automaton minimization techniques based on the use of various simulation relations (e.g., [20, 21, 22, 25, 34, 62]).

In this section, we shall mainly concentrate on the redundancy analysis of initial transitions of linear alternating automata due to the above positive effect this may have on the translation if applied as soon as possible to each automaton built during the translation. Although this specific choice of focus is obviously not the most general strategy for transition simplification, it has, besides special cases that are easier to check than the general problem, advantages that also ease the implementation of the overall translation procedure. For example, restricting the redundancy analysis to the transitions leaving the initial state of the automaton will trivially preserve the correspondence between subformulas of the formula under translation with subautomata of the automaton. Therefore the automata for these subformulas remain directly accessible in case they are needed again during the translation if some of these subformulas occur multiple times in the formula. Another consequence of the restriction is that no transition of the final automaton will be checked twice for redundancy during the translation.

We note, however, that the chosen local strategy of removing transitions incrementally during the translation is in the general case suboptimal in comparison to a global transition redundancy analysis applied to the final automaton. The translation procedure is not "modular" in the sense that the nonredundant transitions of an automaton would always remain so in all automata built from it incrementally. For example, the automaton built for 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 translation rules has redundant (non-initial) transitions even though none of the subautomata rooted at its non-initial states contain any redundant transitions (see Fig. 4.13). Of course, this phenomenon is hardly surprising because of the obvious analogy between incremental transition redundancy analysis and the problem of simplifying a compound LTL formula built from one or two arbitrary subformulas and a connective.

4.4.1 Redundant Transitions and Language Containment

It is clear from the definition of transition redundancy that each redundancy test is equivalent to two language containment checks between alternating automata. That is, if $\mathcal{A}$ and $\mathcal{A}'$ are two alternating automata, where $\mathcal{A}'$ is obtained from $\mathcal{A}$ by removing a transition from it as described above, then the removed transition is redundant iff

$$\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}') \qquad \text{and} \qquad \mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A})$$

hold for the automata. It is easy to see from the definition of $\mathcal{A}'$ that the second condition actually holds trivially, since all fin-accepting runs of $\mathcal{A}'$

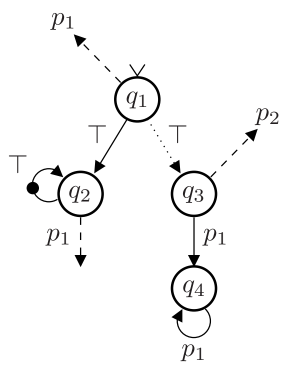


Fig. 4.13: A linear alternating automaton $\mathcal{A}$ built for the formula $(\mathsf{F}p_1 \vee \mathsf{X}(\mathsf{G}p_1 \vee p_2))$ using the original translation rules and then simplifying the guards of transitions. The transition from q_3 and q_4 is redundant in $\mathcal{A}$ even though it is not redundant in $\mathcal{A}^{q_3}$

are always also fin-accepting runs of $\mathcal{A}$. Therefore, it is sufficient to check only for the first condition between the automata to determine whether a given transition is redundant. By the classic reformulation of language containment, this involves finding an automaton for the language $\overline{\mathcal{L}_{\text{fin}}(\mathcal{A}')}$. In previous discussion, we have been able to build such an automaton by exploiting the fact that both automata involved in the test have been built for subformulas of the formula to be translated into an automaton. However, the above language containment problem differs from all previous cases in that the automaton $\mathcal{A}'$ is now obtained directly from another automaton instead of a known LTL formula via formula translation. 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.

Because $\mathcal{A}'$ is still a linear alternating automaton, we can find its corresponding LTL formula ψ, for example, via the reverse translation discussed in Sect. 3.5. In principle, we may then build an automaton for the complemented language $\overline{\mathcal{L}_{\text{fin}}(\mathcal{A}')}$ as before by applying the basic translation to the positive normal form of $\neg\psi$. Unfortunately, as noted in the discussion at the end of Sect. 3.5, a reverse translation based on the systematic application of a simple pattern, such as the one given in Lemma 3.5.1, may result in an LTL formula that cannot be translated succinctly back into an alternating automaton using the basic translation procedure, and the same holds for the negation of the formula. It is nevertheless not always necessary to apply the reverse translation to the whole automaton $\mathcal{A}'$: because $\mathcal{A}$ and $\mathcal{A}'$ are linear alternating automata that differ only in the transitions leaving the start state q of the transition that was removed from $\mathcal{A}$, $\mathcal{A}$ and $\mathcal{A}'$ share all subautomata that do not include the state q. If the automaton $\mathcal{A}$ was built from an LTL formula φ using the basic translation, we can therefore 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 $\mathcal{A}'$. When restricting the redundancy analysis to the initial transitions of $\mathcal{A}$, this implies that the formula ψ can be found from $\mathcal{A}'$ with a single reverse translation step applied to the initial state of the automaton.

Even though we may be able to reuse some subformulas of φ when building the LTL formula ψ from the automaton $\mathcal{A}'$, we are not always able to reuse automata built for these subformulas when translating the positive normal form of $\neg\psi$ into an automaton. This is partly caused by the fact that the formula ψ needs to be negated before the translation, which effectively necessitates building automata for negated subformulas of φ. 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 worst-case number of formula translation subproblems that may arise during the construction and simplification of a linear 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 actually 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 limited transition redundancy analysis while keeping the formula translation subproblems within the set of subformulas of φ and (the positive normal forms of) their negations.

As noted in Sect. 4.2, we could alternatively find an automaton for the complement of the language of $\mathcal{A}'$ by applying to $\mathcal{A}'$ a direct dualization construction designed to support the generalized definition for the automata. However, with our explicit definition for alternating automata, all means of finding a linear alternating automaton accepting the complement of a language recognized by another automaton are bound to cause an exponential blow-up in the number of transitions in some cases (for example, when complementing automata corresponding to LTL formulas, the negations of which are in conjunctive normal form as mentioned in Sect. 4.2).

4.4.2 Special Cases

In this section, we present special cases that allow the detection of redundant transitions in linear alternating automata without solving directly the language containment problem presented in Sect. 4.4.1. These tests combine local structural analysis of the automata with language containment checks that can again be handled by applying the basic translation only to subformulas of a given LTL formula or their negations. Thus, we can in these special cases avoid the reverse translation or dualization of linear alternating automata; however, we may sometimes have to trade the single language containment test of the previous section for possibly several tests that reduce to checking the emptiness of the intersection of more than two languages.

Rephrasing the language containment test of the previous section as a condition on runs of the automaton, we find that a transition is redundant iff each accepting run of $\mathcal{A}$ including an edge labelled with the transition implies the existence of another accepting run (on the same input) in which $\mathcal{A}$ avoids taking this transition. This intuition forms the basic strategy of proving the main results of this section by modifying accepting runs of $\mathcal{A}$ including a given transition into accepting runs that do not contain this transition.

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 Sect. 4.1.2 for removing transitions with an empty guard (characterized by an unsatisfiable Boolean formula in $PL(AP)$ in automata over the alphabet 2^{AP}) from an automaton. Additionally, every automaton will naturally avoid taking any transition, the occurrence of which in a run of the automaton would lead to a subgraph of the run that cannot be a part of any accepting run of the automaton. For example, this occurs if it is impossible to exhibit fin-accepting runs of all subautomata rooted at the target states of a transition regardless of the remaining input. In terms of languages, this implies that the intersection of the languages accepted by the subautomata is empty.

Proposition 4.4.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton. Let $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ for some $q \in Q$, $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$ such that $\bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) = \emptyset$. Then, no fin-accepting run of $\mathcal{A}$ on any $w \in \Sigma^\omega$ contains an edge labelled with the transition t, and thus t is redundant.*

Proof: Let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}$ on some $w \in \Sigma^\omega$, and let $e \in E \cap (V_i \times 2^{V_{i+1}})$ for some $0 \leq i < \omega$ such that $L(e) = t$. Because G is a run, the target states of this edge are labelled with the target states of t. Therefore, by Proposition 2.3.9, $\mathcal{A}^{q'}$ fin-accepts w^{i+1} for all $q' \in Q'$, that is, $w^{i+1} \in \bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$. But then $\bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \neq \emptyset$, which is a contradiction, and thus e cannot be labelled with the transition t. $\square$

Clearly, all redundant transitions of linear alternating automata do not necessarily fall into the above category, and the automaton may well have accepting runs containing edges labelled with a redundant transition. In the following, we shall investigate conditions under which these runs can be modified into accepting runs in which the automaton avoids taking a given transition. At the formal level, we shall often make use of the following result that describes an obvious way to extract a partial run avoiding a given transition from any run of the automaton by truncating each individual path of the run at the first occurrence of an edge labelled with the transition. If we can then show that the partial run can always be extended back into an accepting run (on the same input) that still avoids the transition, it follows that the transition is redundant.

Lemma 4.4.2 *Let $G = \langle V, E, L \rangle$ be a fin-accepting run of the 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 start 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 partial fin-accepting run of $\mathcal{A}$ on w such that none of the edges of E' is labelled with the transition t, and each node of G' with no outgoing edges is labelled with q.

Proof: It is obvious that $V' \subseteq V$ (with $V'_i \subseteq V_i$ for all $0 \leq i < \omega$) and $E' \subseteq E$. This immediately shows that G' can be partitioned into finite disjoint levels with edges between successive levels of G'. Furthermore, $L'(v_0) = L(v_0) = q_I$, because G is a run of $\mathcal{A}$.

Let $v \in V'$. Because G is a run and $V' \subseteq V$, v has a unique outgoing edge $e \in E$ in G whose labelling is consistent with the requirements of a run of $\mathcal{A}$. Because $E' \subseteq E$, v now has either no outgoing edges in G' (if $e \notin E'$), or v keeps its unique outgoing edge also in G'; by the definition of E', this edge is always labelled with a transition different from t. Because $L'(x) = L(x)$ for all $x \in V' \cup E'$, the labelling 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 $L(e) \neq t$. Therefore $e \in E'$, and G' is a partial run of $\mathcal{A}$ on w.

Clearly, because each acceptance sequence $r' \in \mathcal{E}(G')$ is also an acceptance sequence of G and $L'(e) = L(e)$ for all $e \in E'$, $\text{fin}(r') = \emptyset$ holds both in G and G', and thus G' is a partial fin-accepting run of $\mathcal{A}$.

Finally, if $v \in V'$ has no outgoing edges, then the unique edge starting from v in G is necessarily labelled with the transition t, and because the labelling of G is consistent, v is labelled with the start state of t in G, i.e., $L(v) = q = L'(v)$. □

Transition Simulation

Obviously, an accepting run of a linear alternating automaton can be modified to avoid a given transition, for example, if the automaton has another transition that can be substituted systematically for each occurrence of the given transition in the run. Clearly, such a substitution can be made only if both transitions have the same start state, and if the automaton could in fact have taken either of the transitions at each occurrence of the given transition in the run. A strict transition substitution by a simple relabelling of edges in the run obviously requires that the transitions also share their target states. Clearly, a less restrictive strategy for transition substitution is to confirm only that the choice between the transitions does not affect the ability of the subautomata rooted at the target states of the respective transitions to accept the remainder of the input. This intuition leads to the traditional notion of checking for *simulations* between transitions.

Formally, a transition $t_1 = \langle q_1, \Gamma_1, F_1, Q'_1 \rangle \in \Delta$ *simulates* another transition $t_2 = \langle q_2, \Gamma_2, F_2, Q'_2 \rangle \in \Delta$ in an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ iff $q_1 = q_2$, $\Gamma_2 \subseteq \Gamma_1$, and $\bigcap_{q' \in Q'_2} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \subseteq \bigcap_{q' \in Q'_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$. (We call t_2 the *simulated* transition and t_1 the *simulating* transition.) By the above observations, the existence of a transition that simulates another (different) transition suggests that the other transition may possibly be removed from the automaton. However, this can be done only if the simulation relationship between the transitions does not depend on the simulated transition, i.e., only if $\bigcap_{q' \in Q'_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ still includes $\bigcap_{q' \in Q'_2} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ after the simulated transition is removed from the automaton. Clearly, this language containment relationship may cease to hold if the simulated transition starts from a state that is reachable from some $q' \in Q'_1$ in the original automaton. In the class of linear alternating automata, however, the language containment relationship is easily seen to be preserved if the simulating transition is not a self-loop.

Proposition 4.4.3 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton that contains a transition $t_1 = \langle q, \Gamma_1, F_1, Q_1' \rangle \in \Delta$ and a non-self-loop transition $t_2 = \langle q, \Gamma_2, F_2, Q_2' \rangle \in \Delta$, $t_1 \neq t_2$, $q \notin Q_2'$, such that $\Gamma_1 \subseteq \Gamma_2$, and $\bigcap_{q' \in Q_1'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \subseteq \bigcap_{q' \in Q_2'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$. Define the automaton $\mathcal{A}' \stackrel{\text{def}}{=} \langle \Sigma, Q, \Delta \setminus \{t_1\}, q_I, \mathcal{F} \rangle$ obtained from $\mathcal{A}$ by removing the transition t_1 from Δ. Then, $\mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big) = \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$.*

Proof: As noted in Sect. 4.4.1, $\mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big) \subseteq \mathcal{L}_{\text{fin}}\big(\mathcal{A}^q\big)$ holds trivially. For the other direction, let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$. By Lemma 4.4.2, we can extract from a fin-accepting run of $\mathcal{A}^q$ on w a partial fin-accepting run $G' = \langle V', E', L' \rangle$ that contains no edges labelled with the transition t_1. This obviously makes G' also a partial fin-accepting run of $(\mathcal{A}')^q$ 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 labelled with the transition t_1. Because G is fin-accepting, $w(i) \in \Gamma_1$, and the union of the labels of the successors of v in G is equal to Q_1', which implies (Proposition 2.3.9) that $(\mathcal{A}^q)^{q'}$ $(= \mathcal{A}^{q'})$ fin-accepts w^{i+1} for all $q' \in Q_1'$, i.e., $w^{i+1} \in \bigcap_{q' \in Q_1'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$.

From the assumptions it now follows that $w(i) \in \Gamma_2$ (because $\Gamma_1 \subseteq \Gamma_2$) and $w^{i+1} \in \bigcap_{q' \in Q_2'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$, and therefore $\mathcal{A}^{q'}$ $(= (\mathcal{A}^q)^{q'})$ fin-accepts w^{i+1} also for all $q' \in Q_2'$. By Lemma 3.4.1, this implies that $\mathcal{A}^q$ has a fin-accepting run on w^i, where the initial edge of this run is labelled with the transition t_2, and the target nodes of this edge are labelled with the states of Q_2'. But then, because $\mathcal{A}^q$ is a linear alternating automaton and $q \notin Q_2'$, it follows that no other edge in this run can be labelled with the transition t_1, either, and thus 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, G' can be extended into a fin-accepting run of $(\mathcal{A}')^q$ on w by Proposition 2.3.10, and thus $w \in \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big)$ and $\mathcal{L}_{\text{fin}}(\mathcal{A}^q) \subseteq \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big)$. □

If the simulating transition is a self-loop, it includes its own start state in its target states. Because also the simulated transition is rooted at this state, the simulation relationship between the transitions may thus depend on the simulated transition, and thus removing the simulated transition from the automaton may change the language of the subautomaton rooted at the common start state of the transitions. However, we may still not need to resort to, for example, reverse translation based transition redundancy analysis if also the simulated transition is a self-loop.

Consider an acceptance sequence in a fin-accepting run of the automaton. If the start state of the simulated self-loop is a transient state of the automaton, then this self-loop can occur only finitely many times in the sequence (cf. Corollary 2.3.16). Because also the simulating transition is a self-loop, it cannot contribute to the acceptance conditions occurring infinitely many times in the transitions along this sequence, either. The requirements of transition simulation now guarantee that the language of the subautomaton rooted at the transient state is preserved when the simulated self-loop is removed.

In case the start state of the simulated self-loop is not a transient state in the automaton, simulation is not by itself sufficient to allow the self-loop to be removed from the automaton if some acceptance sequence in a run of the automaton converges to this nontransient state. Because simulation does not depend on the acceptance conditions of the transitions, it may occur that the nontransient start state of the simulated self-loop becomes transient in the modification of the automaton. This may then change the language of the automaton, for example, if the new transient state has no other successors than itself. Obviously, this problem will not emerge if the simulated self-loop includes all acceptance conditions of the simulating self-loop in its acceptance conditions. This requirement can even be generalized slightly: the simulated self-loop can safely be removed if there exists a *set* of simulating self-loops that share their *common* acceptance conditions with the simulated self-loop. (In linear alternating automata, the acceptance conditions of these self-loops can affect fin-acceptance only in acceptance sequences converging to the common start state of the transitions. Therefore, for example, in an acceptance sequence with an infinite suffix in which the automaton simply repeats the simulated transition, each occurrence of this transition can be replaced with a permutation of the simulating self-loops. By the choice of these self-loops, none of the acceptance conditions not associated with the simulated self-loop will then occur in all transitions in this self-loop permutation.)

The above informal discussion can be summarized as the following result.

Proposition 4.4.4 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton that contains a self-loop $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ for some $q \in Q, \Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q' \subseteq Q$ ($q \in Q'$), together with a nonempty finite set of self-loops $\widetilde{\Delta} = \{t_1, \ldots, t_{|\widetilde{\Delta}|}\} \subseteq \Delta \setminus \{t\}$ such that for all $t_i = \langle q, \widetilde{\Gamma}_i, \widetilde{F}_i, \widetilde{Q}'_i \rangle \in \widetilde{\Delta}$ ($1 \leq i \leq |\widetilde{\Delta}|$), $\Gamma \subseteq \widetilde{\Gamma}_i$, and $\bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \subseteq \bigcap_{q' \in \widetilde{Q}'_i} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$. Define 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. Then, $\mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big) = \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ if q is a transient state of $\mathcal{A}$, or if $\bigcap_{1 \leq i \leq |\widetilde{\Delta}|} \widetilde{F}_i \subseteq F$.*

Proof: As noted in Sect. 4.4.1, $\mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big) \subseteq \mathcal{L}_{\text{fin}}\big(\mathcal{A}^q\big)$ holds trivially. We thus check the other direction. Let $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$, and let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}^q$ on w. We again use Lemma 4.4.2 to extract from G' a partial fin-accepting run G' of $(\mathcal{A}')^q$ on w and show that G' can always be extended into a fin-accepting run of $(\mathcal{A}')^q$ on w under either of the conditions given in the proposition. Let thus $v \in V'_i$ be a node with no outgoing edges in G' for some $0 \leq i < \omega$; by Lemma 4.4.2, $L'(v) = L(v) = q$. We construct a fin-accepting run of $\big((\mathcal{A}')^q\big)^{L'(v)} = \big((\mathcal{A}')^q\big)^q = (\mathcal{A}')^q$ on w^i; because the same construction applies to all nodes of G' with no outgoing edges, the result then follows by applying Proposition 2.3.10.

Because $L(v) = q$, v is 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 sequence of consecutive edges $(e_k)_{0 \leq k < j+1}$, $e_k \in E \cap (V_{i+k} \times 2^{V_{i+k+1}})$, where $L(e_k) = \langle q, \Gamma_k, F_k, Q'_k \rangle \in \Delta$ is an initial self-loop of $\mathcal{A}^q$ for all $0 \leq k < j$, and if $j < \omega$, then $L(e_j) = \langle q, \Gamma_j, F_j, Q'_j \rangle \in \Delta$ is an initial transition of $\mathcal{A}^q$ that is not a self-loop. We shall modify this edge sequence into a partial fin-accepting run of $(\mathcal{A}')^q$ on w^i by replacing all edges labelled with the self-loop t in this sequence systematically with edges labelled with self-loops in $\widetilde{\Delta}$. A simple strategy for choosing the labels of the replacement edges is to cycle through the transitions of $\widetilde{\Delta}$ repeatedly such that the edges of $(e_k)_{0\leq k<j+1}$ labelled with t are successively replaced with edges labelled with the transitions t_1, t_2, …, $t_{|\widetilde{\Delta}|}$, followed again by another round through elements of $\widetilde{\Delta}$ if necessary. Formally, if $L(e_k) = t$ for some $0 \leq k < j+1$, then e_k will be replaced with an edge labelled with $t_{\lambda(k)}$, where the function $\lambda : \{l \in \mathbb{N} \,|\, 0 \leq l < j+1\} \to \{1, 2, \ldots, |\widetilde{\Delta}|\}$ is defined by the rule $\lambda(k) \stackrel{def}{=} 1 + \big(|\{0 \leq l < k \mid L(e_l) = t\}| \mod |\widetilde{\Delta}|\big)$. This labelling strategy guarantees a "fair" distribution for the new labels such that if t occurs infinitely many times as the label of some edge in the original sequence, then also each $t' \in \widetilde{\Delta}$ will occur infinitely often as the label of some edge in the modified edge sequence.

To ensure that the labelling can be made consistent with a partial fin-accepting run of $(\mathcal{A}')^q$, the number of successors of the edge replacing e_k in the original sequence must be chosen to match the number of distinct elements in the target state set of $t_{\lambda(k)}$. For this purpose, we first write $Q'_k \setminus \{q\} = \{q_{k,1}, \ldots, q_{k,n_k}\}$ for some $0 \leq n_k < \omega$, and $\widetilde{Q}'_l \setminus \{q\} = \{\tilde{q}_{l,1}, \ldots, \tilde{q}_{l,|\widetilde{Q}'_l|-1}\}$ for all $0 \leq k < j+1$ and $1 \leq l \leq |\widetilde{\Delta}|$, respectively. We then let

$$m_k \stackrel{def}{=} \begin{cases} n_k & \text{if } L(e_k) \neq t \\ |\widetilde{Q}'_{\lambda(k)}| - 1 & \text{otherwise} \end{cases}$$

for all $0 \leq k < j+1$ and use the constants m_k to define the graph $\widehat{G} \stackrel{def}{=} \langle \widehat{V}, \widehat{E}, \widehat{L} \rangle$, where

- $\widehat{V}_0 \stackrel{def}{=} \{\hat{v}_0\}$, $\widehat{V}_{k+1} \stackrel{def}{=} \{v_{k,1}, \ldots, v_{k,m_k}\} \cup \begin{cases} \{\hat{v}_k\} & \text{if } k < j \\ \emptyset & \text{otherwise} \end{cases}$ for all $0 \leq k < j+1$, and if $j < \omega$, then $\widehat{V}_k \stackrel{def}{=} \emptyset$ for all $j+1 < k < \omega$;
- $\widehat{E} \stackrel{def}{=} \bigcup_{0\leq k<j+1} \big\{\langle \hat{v}_k, \widehat{V}_{k+1} \rangle\big\}$; and
- for all $0 \leq k < j+1$, $\widehat{L}(\hat{v}_k) \stackrel{def}{=} q$, $\widehat{L}(v_{k,l}) \stackrel{def}{=} \begin{cases} q_{k,l} & \text{if } L(e_k) \neq t \\ \tilde{q}_{\lambda(k),l} & \text{otherwise} \end{cases}$ for all $1 \leq l \leq m_k$, and $L\big(\langle \hat{v}_k, \widehat{V}_{k+1} \rangle\big) \stackrel{def}{=} \begin{cases} L(e_k) & \text{if } L(e_k) \neq t \\ t_{\lambda(k)} & \text{otherwise} \end{cases}$.

$\widehat{G}$ is a partial run of $(\mathcal{A}')^q$ on w^i:

- $\widehat{V}_0 = \{\hat{v}_0\}$, $\widehat{L}(\hat{v}_0) = q$, and $\widehat{G}$ is partitioned into finite disjoint levels with edges between successive levels of $\widehat{G}$.
- Let $v' \in \widehat{V}_k$ for some $0 \leq k < j+1$. Then v' either has no outgoing edges, or $v' = \hat{v}_k$, $\widehat{L}(v') = q$, and v' has the unique outgoing edge $e = \langle v', \widehat{V}_{k+1} \rangle \in \widehat{E}$ for which either $\widehat{L}(e) = L(e_k) = \langle q, \Gamma_k, F_k, Q'_k \rangle \in \Delta'$, or $\widehat{L}(e) = t_{\lambda(k)} = \langle q, \widetilde{\Gamma}_{\lambda(k)}, \widetilde{F}_{\lambda(k)}, \widetilde{Q}'_{\lambda(k)} \rangle \in \Delta'$.

  If $L(e) = L(e_k)$, then, because $e_k \in V_{i+k} \times 2^{V_{i+k+1}}$ is an edge in the fin-accepting run G, it follows that $w(i+k) = (w^i)(k) \in \Gamma_k$. Furthermore, the definition of $\widehat{L}$ guarantees that $\widehat{L}(\widehat{V}_{k+1}) = Q'_k$.

  Otherwise, if $L(e) \neq L(e_k)$, then $L(e_k) = t$, $\Gamma_k = \Gamma$, and $(w^i)(k) \in \Gamma$. Because $\Gamma \subseteq \widetilde{\Gamma}_l$ holds for all $1 \leq l \leq |\widetilde{\Delta}|$, it follows that $(w^i)(k) \in$

$\widetilde{\Gamma}_{\lambda(k)}$. It is again easy to check that $\widehat{L}(\widehat{V}_{k+1}) = \widetilde{Q}'_{\lambda(k)}$. Thus the edge labelling $\widehat{L}$ is consistent.

- By the definition of $\widehat{E}$, each node $v' \in \widehat{V}_k$ for some $1 \leq k < \omega$ is obviously a successor of the node $\hat{v}_k \in \widehat{V}_{k-1}$ in $\widehat{G}$.

If $j < \omega$, then $\widehat{E}$ contains only finitely many edges, and thus $\widehat{G}$ is trivially a partial fin-accepting run of $(\mathcal{A}')^q$ on w^i. By the choice of j, this occurs whenever q is a transient state of $\mathcal{A}$: otherwise the sequence of edges $(e_k)_{0\leq k<j+1}$ labelled with initial self-loops of $\mathcal{A}^q$ would violate the fin-acceptance condition, which is impossible, because the sequence was extracted from a fin-accepting run of $\mathcal{A}^q$ on w^i.

If $j = \omega$, then the run $\widehat{G}$ contains the unique acceptance sequence $(\hat{e}_k)_{0\leq k<\omega} = \big(\langle \hat{v}_k, \widehat{V}_{k+1}\rangle\big)_{0\leq k<\omega}$. If $L(e_k) = t$ holds for only finitely many $0 \leq k < \omega$ in the original sequence, then there exists an index $0 \leq l < \omega$ such that $\widehat{L}(\hat{e}_k) = L(e_k)$ holds for all $l \leq k < \omega$ by the definition of $\widehat{G}$. In this case $\mathrm{fin}\big((\hat{e}_k)_{0\leq k<\omega}\big) = \mathrm{fin}\big((\hat{e}_k)_{l\leq k<\omega}\big) = \mathrm{fin}\big((e_k)_{l\leq k<\omega}\big) = \mathrm{fin}\big((e_k)_{0\leq k<\omega}\big) = \emptyset$, and thus $\widehat{G}$ is a partial fin-accepting run of $(\mathcal{A}')^q$ on w^i.

Otherwise $(e_k)_{0\leq k<\omega}$ contains infinitely many edges labelled with the transition t. Let $\tau(x) \stackrel{def}{=} \{t' \in \Delta \mid x(k) = t' \text{ for infinitely many } 0 \leq k < \omega\}$ be the set of transitions occurring infinitely many times in an infinite sequence of transitions $x \in \Delta^\omega$. Using this definition, we write $T \stackrel{def}{=} \tau\big((L(e_k))_{0\leq k<\omega}\big)$ and $\widehat{T} \stackrel{def}{=} \tau\big((\widehat{L}(\hat{e}_k))_{0\leq k<\omega}\big)$ as the sets of transitions occurring infinitely many times as labels of edges in $(e_k)_{0\leq k<\omega}$ and $(\hat{e}_k)_{0\leq k<\omega}$, respectively.

Because $t \in T$, it now follows from the definition of $\widehat{G}$ that $\widetilde{\Delta} \subseteq \widehat{T}$. We show that for all acceptance conditions $f \in \mathcal{F}$, $\widehat{T}$ now contains a transition that does not include f in its acceptance conditions. Because $(\hat{e}_k)_{0\leq k<\omega}$ then contains infinitely many edges labelled with this transition, it follows that $f \notin \mathrm{fin}\big((\hat{e}_k)_{0\leq k<\omega}\big)$. By quantifying over all acceptance conditions in $\mathcal{F}$, we can then conclude that $\mathrm{fin}\big((\hat{e}_k)_{0\leq k<\omega}\big) = \emptyset$, and $\widehat{G}$ is a partial fin-accepting run of $(\mathcal{A}')^q$ on w^i.

If $f \in F$ (i.e., if t includes f in its acceptance conditions), then, because $t \in T$ and $\mathrm{fin}\big((e_k)_{0\leq k<\omega}\big) = \emptyset$, T necessarily contains a transition $t' \in \Delta\setminus\{t\}$ that does not include f in its acceptance conditions. Therefore $L(e_k) = t' \neq t$ for infinitely many $0 \leq k < \omega$, which implies that $\widehat{L}(\hat{e}_k) = L(e_k) = t'$ also holds for infinitely many k by the definition of $\widehat{G}$. It follows that $t' \in \widehat{T}$ and $f \notin \mathrm{fin}\big((\hat{e}_k)_{0\leq k<\omega}\big)$.

Let $f \in \mathcal{F}\setminus F$. If $f \in \mathrm{fin}\big((\hat{e}_k)_{0\leq k<\omega}\big)$, then there exists an index $0 \leq l < \omega$ such that for all $l \leq k < \omega$, $\widehat{L}(\hat{e}_k) \in \widehat{T}$, and f is included in the acceptance conditions of $\widehat{L}(\hat{e}_k)$. In particular, because $\widetilde{\Delta} \subseteq \widehat{T}$, f is included in the acceptance conditions of each transition $t' \in \widetilde{\Delta}$, i.e., $f \in \bigcap_{1\leq m\leq|\widetilde{\Delta}|} \widetilde{F}_m$. But then, because $\bigcap_{1\leq m\leq|\widetilde{\Delta}|} \widetilde{F}_m \subseteq F$, $f \in F$, which is a contradiction, and therefore $\widehat{T}$ necessarily contains a transition not including f in its acceptance conditions. We conclude that $\widehat{G}$ is a partial fin-accepting run of $(\mathcal{A}')^q$ on w^i.

We finally extend $\widehat{G}$ into a fin-accepting run of $(\mathcal{A}')^q$ on w^i. Let $v' \in \widehat{V}_k$ for some $0 \leq k < \omega$ be a node with no outgoing edges in $\widehat{G}$. Then $k \geq 1$, v' is one of the nodes $v_{k-1,l}$ for some $1 \leq l \leq m_{k-1}$, and $\widehat{L}(v') \neq q$. Because the fin-accepting run G contains the edge $e_{k-1} \in V_{i+k-1} \times 2^{V_{i+k}}$, the target states of which are labelled with (exactly) the states in Q'_{k-1}, it follows that $(\mathcal{A}^q)^{q'}$ $(= \mathcal{A}^{q'})$ fin-accepts $w^{i+k} = (w^i)^k$ for all $q' \in Q'_{k-1}$, i.e., $(w^i)^k \in \bigcap_{q' \in Q'_{k-1}} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$.

We show that $(w^i)^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{\widehat{L}(v')})$. This is clear if $L(e_{k-1}) \neq t$, because then $\widehat{L}(v') = q_{k-1,l} \in Q'_{k-1}$ holds by the definition of $\widehat{G}$. Otherwise $Q'_{k-1} = Q'$, and thus $(w^i)^k \in \bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$. The assumption that $\bigcap_{q' \in Q'} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'}) \subseteq \bigcap_{q' \in \widetilde{Q}'_m} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ holds for all $1 \leq m \leq |\widetilde{\Delta}|$ now implies that $(w^i)^k \in \mathcal{L}_{\text{fin}}(\mathcal{A}^{q'})$ for all $q' \in \bigcup_{1 \leq m \leq |\widetilde{\Delta}|} \widetilde{Q}'_m$. The result then follows from the definition of $\widehat{G}$, since $\widehat{L}(v') = \tilde{q}_{\lambda(k-1),l} \in \bigcup_{1 \leq m \leq |\widetilde{\Delta}|} \widetilde{Q}'_m$ holds in this case.

Because $\widehat{L}(v') \neq q$ is a successor of q and $\mathcal{A}$ is a linear alternating automaton, $\mathcal{A}^{\widehat{L}(v')} = (\mathcal{A}^q)^{\widehat{L}(v')} = \big((\mathcal{A}')^q\big)^{\widehat{L}(v')}$, and it follows that $\big((\mathcal{A}')^q\big)^{\widehat{L}(v')}$ fin-accepts $(w^i)^k$. By repeating the same argument for all nodes of $\widehat{G}$ with no outgoing edges, we can extend $\widehat{G}$ into a fin-accepting run of $(\mathcal{A}')^q$ on w^i by Proposition 2.3.10.

Returning to the partial run G', we can now repeat the above construction for all $0 \leq i < \omega$ to find fin-accepting runs of $(\mathcal{A}')^q$ on w^i for all $v \in V'_i$ having no outgoing edges. By Proposition 2.3.10, this implies that G' can be extended into a fin-accepting run of $(\mathcal{A}')^q$ on w, and thus $w \in \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big)$, and $\mathcal{L}_{\text{fin}}(\mathcal{A}^q) \subseteq \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^q\big)$. □

To illustrate the application of Propositions 4.4.3 and 4.4.4, we again consider the example previously discussed in Sect. 4.3.4.

Example 4.4.5 As shown in Sect. 4.3.4, it is possible to build a single-state automaton for the LTL formula $\varphi \stackrel{def}{\equiv} \Big(\big(\bot \mathsf{R_w} (\top \mathsf{U_s} p_1)\big) \wedge \big(\bot \mathsf{R_w} (\top \mathsf{U_s} p_2)\big)\Big)$ using the refined translation rules given in Sect. 4.3.3 (Fig. 4.14 (a)). Let q denote the single state of this automaton. The transition relation of the automaton consists of the self-loops $t_1 = \langle q, \top, \{f_1, f_2\}, \{q\} \rangle$, $t_2 = \langle q, p_1, \{f_2\}, \{q\} \rangle$, $t_3 = \langle q, p_2, \{f_1\}, \{q\} \rangle$ and $t_4 = \langle q, (p_1 \wedge p_2), \emptyset, \{q\} \rangle$ (where f_1 and f_2 correspond to the acceptance conditions denoted in Fig. 4.14 by • and ◦, respectively).

It is easy to see that the guard of t_4 implies the guards of the transitions t_2 and t_3. Furthermore, because the empty intersection of the acceptance conditions of t_2 and t_3 is trivially a subset of the acceptance conditions of t_4, the preconditions of Proposition 4.4.4 are satisfied with $t = t_4$ and $\widetilde{\Delta} = \{t_2, t_3\}$ (because all transitions have the same target state, the language containment relationships hold trivially). Thus the transition t_4 can be removed from the automaton as shown in Fig. 4.14 (b).

Joining the automaton in Fig. 4.14 (b) with the automaton shown in Fig. 4.14 (c) (built for the formula $\psi \stackrel{def}{\equiv} \big(p_3 \mathsf{R_w} (p_4 \mathsf{R_s} p_5)\big)$ in Sect. 4.3.4) using the rule given for the $\vee$ connective now results in the automaton shown in Fig. 4.14 (d) for the formula $(\varphi \vee \psi)$. We can now apply Proposition 4.4.3 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 $\top$-labelled initial non-self-loop of the automaton simulates the p_2-labelled initial transi-

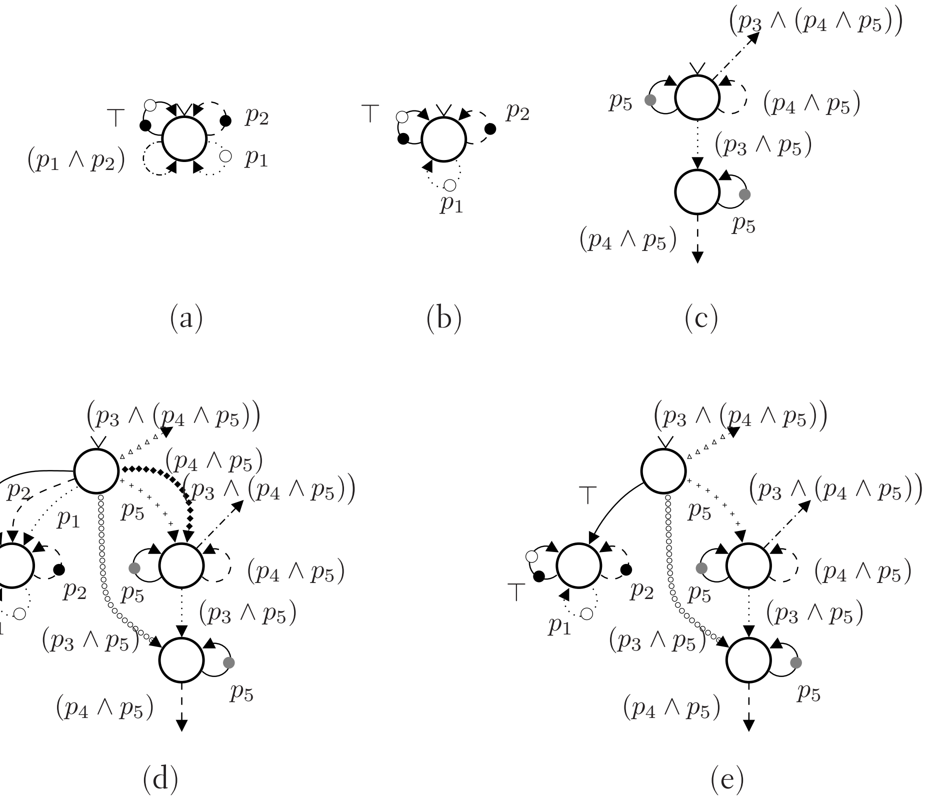


Fig. 4.14: Using Propositions 4.4.3 and 4.4.4 to improve formula translation. (a) Automaton built for the formula $\varphi \overset{def}{\equiv} \big((\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_1)) \wedge (\bot \,\mathsf{R_w}\, (\top \,\mathsf{U_s}\, p_2)) \big)$ in Sect. 4.3.4; (b) Automaton obtained from (a) by the application of Proposition 4.4.4; (c) Automaton built for the formula $\psi \overset{def}{\equiv} (p_3 \,\mathsf{R_w}\, (p_4 \,\mathsf{R_s}\, p_5))$ in Sect. 4.3.4; (d) Automaton built from (b) and (c) for the formula $(\varphi \vee \psi)$; (e) Automaton obtained from (d) by repeated application of Proposition 4.4.3

tion, and thus the latter transition can be removed from the automaton by Proposition 4.4.3 (again, the language containment relationship holds trivially). It can be checked that Propositions 4.4.3 and 4.4.4 do not allow removing more transitions from this automaton. ■

In the above example, the language containment relationships required for applying Propositions 4.4.3 and 4.4.4 were easy to establish. In the general case, both propositions specify a condition on language containment between two languages obtained by intersecting sets of languages. We shall show in the next section how these conditions can be rewritten as multiple simple language containment problems, all of which can be checked during formula translation without a need to apply reverse translation or dualization to the automata. Nevertheless, it is easy to see that the above results do not capture all opportunities for detecting redundant transitions in linear alternating automata. For example, the results do not cover the simulation of non-self-loop transitions with self-loops, which could be applied, for example, to simplify the automaton built for the LTL formula $\big(\mathsf{G}(p_1 \wedge p_2)\,\mathsf{R_w}\,p_1\big) \in LTL\big(\{p_1, p_2\}\big)$ using the basic rules. (The initial state of this automaton is the start state of two transitions, one of which is a self-loop that simulates the other non-self-loop transition. However, neither of the above propositions applies in this case.)

Checking for Containment Between Intersections of Languages

We end Sect. 4.4 with a discussion on solving the language containment problems occurring in the conditions of Propositions 4.4.3 and 4.4.4. In these propositions, the single language containment problem discussed in Sect. 4.4.1 is traded for the more complex condition

$$\bigcap_{q \in Q_1} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) \subseteq \bigcap_{q \in Q_2} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$$

for some subsets $Q_1, Q_2 \subseteq Q$ of states of a linear alternating automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$. This condition corresponds to several instances of the basic problem of checking the emptiness of the intersection of languages recognized by a set of automata; the reformulation is based on basic facts on families of sets. We repeat these facts below for reference.

Lemma 4.4.6 *Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two families of sets. Then,*

$$\bigcap_{X \in \mathcal{X}_1} X \subseteq \bigcap_{X \in \mathcal{X}_2} X \qquad \textit{iff} \qquad \bigcap_{X \in \mathcal{X}_1} X \subseteq \bigcap_{X \in \mathcal{X}_2 \setminus \mathcal{X}_1} X.$$

Proof: ("$\Rightarrow$") Assume that $\bigcap_{X \in \mathcal{X}_1} X \subseteq \bigcap_{X \in \mathcal{X}_2} X$, and let $x \in \bigcap_{X \in \mathcal{X}_1} X$. Then $x \in \bigcap_{X \in \mathcal{X}_2} X$, i.e., $x \in X$ for all $X \in \mathcal{X}_2$. Then obviously $x \in X$ also for all $X \in \mathcal{X}_2 \setminus \mathcal{X}_1$, and thus $x \in \bigcap_{X \in \mathcal{X}_1 \setminus \mathcal{X}_2} X$ and $\bigcap_{X \in \mathcal{X}_1} X \subseteq \bigcap_{X \in \mathcal{X}_2 \setminus \mathcal{X}_1} X$.

("$\Leftarrow$") Let $\bigcap_{X \in \mathcal{X}_1} X \subseteq \bigcap_{X \in \mathcal{X}_2 \setminus \mathcal{X}_1} X$, and let $x \in \bigcap_{X \in \mathcal{X}_1} X$. Then $x \in X$ for all $X \in \mathcal{X}_1$, and $x \in \bigcap_{X \in \mathcal{X}_2 \setminus \mathcal{X}_1} X$, i.e., $x \in X$ for all $X \in \mathcal{X}_2 \setminus \mathcal{X}_1$. But then $x \in X$ for all $X \in \mathcal{X}_1 \cup (\mathcal{X}_2 \setminus \mathcal{X}_1) = \mathcal{X}_1 \cup \mathcal{X}_2$. Thus especially $x \in X$ for all $X \in \mathcal{X}_2$, and it follows that $x \in \bigcap_{X \in \mathcal{X}_2} X$ and $\bigcap_{X \in \mathcal{X}_1} X \subseteq \bigcap_{X \in \mathcal{X}_2} X$. □

Lemma 4.4.6 allows us to discard the sets common to both families from the intersection on the right hand side of the set inclusion equation. Using this result, the set inclusion problem can be written in a form that leads to an alternative formulation for the language containment problem.

Lemma 4.4.7 *Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two families of subsets of a set S. Then,*

$$\bigcap_{X\in\mathcal{X}_1} X \subseteq \bigcap_{X\in\mathcal{X}_2} X \qquad \textit{iff} \qquad \overline{X_2} \cap \bigcap_{X\in\mathcal{X}_1} X = \emptyset \quad \textit{for all } X_2 \in \mathcal{X}_2 \setminus \mathcal{X}_1$$

(where $\overline{X} \overset{def}{=} S \setminus X$ for all $X \subseteq S$).

Proof:

$$\begin{array}{lll}
 & \bigcap_{X\in\mathcal{X}_1} X \subseteq \bigcap_{X\in\mathcal{X}_2} X & \\
\text{iff} & \bigcap_{X\in\mathcal{X}_1} X \subseteq \bigcap_{X\in\mathcal{X}_2\setminus\mathcal{X}_1} X & \textit{(Lemma 4.4.6)} \\
\text{iff} & \big(\bigcap_{X\in\mathcal{X}_1} X\big) \cap \overline{\big(\bigcap_{X\in\mathcal{X}_2\setminus\mathcal{X}_1} X\big)} = \emptyset & \\
\text{iff} & \big(\bigcap_{X\in\mathcal{X}_1} X\big) \cap \big(\bigcup_{X\in\mathcal{X}_2\setminus\mathcal{X}_1} \overline{X}\big) = \emptyset & \\
\text{iff} & \bigcup_{X_2\in\mathcal{X}_2\setminus\mathcal{X}_1} \big(\overline{X_2} \cap \bigcap_{X_1\in\mathcal{X}_1} X_1\big) = \emptyset & \\
\text{iff} & \overline{X_2} \cap \bigcap_{X_1\in\mathcal{X}_1} X_1 = \emptyset \text{ for all } X_2 \in \mathcal{X}_2 \setminus \mathcal{X}_1. &
\end{array}$$

□

In the language containment problem, the families of languages (sets) to be intersected are determined by quantifications over the state sets Q_1 and Q_2. By switching the notation, we can rewrite Lemma 4.4.7 to establish the following immediate corollary:

Corollary 4.4.8 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton, and let $Q_1, Q_2 \subseteq Q$ be two subsets of states of $\mathcal{A}$. Then,*

$$\bigcap_{q\in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \subseteq \bigcap_{q\in Q_2} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \qquad \textit{iff} \qquad \overline{\mathcal{L}_{\text{fin}}(\mathcal{A}^{q_2})} \cap \bigcap_{q_1\in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^{q_1}) = \emptyset$$

for all $q_2 \in Q_2 \setminus Q_1$.

A further corollary of the above result is the well-known special case (used, for example, by Gastin and Oddoux in their translation procedure [28]) where the language containment is directly implied by a set inclusion between Q_1 and Q_2; if $Q_2 \subseteq Q_1$, then the right-hand side condition of Corollary 4.4.8 holds trivially.

Corollary 4.4.9 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be an alternating automaton, and let $Q_1, Q_2 \subseteq Q$ be two subsets of states of $\mathcal{A}$. Then,*

$$\bigcap_{q\in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \subseteq \bigcap_{q\in Q_2} \mathcal{L}_{\text{fin}}(\mathcal{A}^q) \qquad \textit{if} \qquad Q_2 \subseteq Q_1.$$

(Intuitively, this result is easy to justify: if $Q_1 \supseteq Q_2$, $w \in \bigcap_{q\in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$ 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 "more difficult" for w to belong to the language $w \in \bigcap_{q\in Q_1} \mathcal{L}_{\text{fin}}(\mathcal{A}^q)$.)

Corollary 4.4.8 reduces the language containment problems in Propositions 4.4.3 and 4.4.4 into multiple questions on language emptiness, where each emptiness check involves one complemented language. When applying these results to the simplification of a linear alternating automaton $\mathcal{A}$ built from an LTL formula, the fact that the language emptiness checks refer to languages (or their complements) accepted by subautomata of $\mathcal{A}$ implies that each language in the emptiness check can be represented by an automaton built for a (possibly negated) subformula of the LTL formula. By postponing the transition redundancy tests after $\mathcal{A}$ has been defined using a translation rule, it is easy to see that automata for the noncomplemented languages are guaranteed to be already available for the emptiness check due to the way in which $\mathcal{A}$ has been built from its subautomata. This holds also if the emptiness check involves the automaton $\mathcal{A}$ itself (i.e., if $q_I \in Q_1$ holds for $\mathcal{A}$'s initial state q_I in Corollary 4.4.8), because the language of $\mathcal{A}$ is preserved even if a redundant transition is later removed from it. Thus each emptiness check requires solving at most one "new" formula translation subproblem to find an automaton for the complemented language. By restricting transition redundancy analysis to cases detectable using Propositions 4.4.1, 4.4.3 and 4.4.4, all language emptiness checks that arise during the translation of an LTL formula into an automaton can be solved with at most $2 \cdot |\mathrm{Sub}(\varphi')|$ formula translation subproblems. This is the same number of subproblems that may arise in the worst case when applying the simplification heuristics presented in Sect. 4.3.

4.5 SUMMARY

The results presented in this chapter suggest repeating the following high-level steps for translating an LTL formula φ into a linear alternating automaton. The translation starts from rewriting φ in positive normal form φ' and building trivial automata for all Boolean constants and atomic subformulas of φ'.

1. Choose a formula $\psi \in \mathrm{Sub}(\varphi')$ that has not yet been translated into an automaton such that automata for all top-level subformulas of ψ have already been constructed.

2. Check whether any of the identities discussed in Sect. 4.3.1 apply to 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, Proposition 4.3.2, Table 4.2) and apply it to the automata built for the top-level subformula(s) of ψ to obtain an automaton $\mathcal{A}_\psi$.

4. Check $\mathcal{A}_\psi$ for redundant transitions (Sect. 4.1, Sect. 4.4.1, Propositions 4.4.1, 4.4.3 and 4.4.4).

5. Repeat from step 1 until $\mathcal{A}_{\varphi'}$ has been constructed.

In principle, the procedure can easily be improved with additional minimization steps, for example, to reduce $\mathcal{A}_\psi$ into a trivial 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 also possible to use language containment tests to replace states systematically with their language-equivalent representatives in each remaining transition as suggested by Rohde [58], or to reduce the number of target states in the transitions.

5 NONDETERMINIZATION OF LINEAR ALTERNATING AUTOMATA

In this chapter we review constructions for translating linear alternating automata into nondeterministic automata. Since nondeterministic automata are known to be equally expressive to alternating automata on both finite and infinite inputs [6, 9, 49], reducing the emptiness checking problem of alternating automata to the corresponding problem on nondeterministic automata allows the use of efficient graph algorithms for emptiness checking.

Due to the succinctness of alternating automata, the size of a nondeterministic automaton corresponding to an alternating automaton may be exponential in the size of the alternating automaton. Consequently, the size of a nondeterministic automaton that recognizes the language of a given LTL formula may be exponential in the size of the formula. This worst-case blow-up does nevertheless not realize for simple syntactically restricted subclasses of LTL, such as for LTL formulas that allow a linear-size translation to alternating automata trivially identifiable with nondeterministic automata. We start our discussion from these special cases in Sect. 5.1; we then review a classic result on the canonicalization of runs of linear alternating automata in Sect. 5.2, to be used in Sect. 5.3, where we shall finally describe a general translation from linear alternating automata to nondeterministic automata, building on the previous ideas of Gastin and Oddoux [28].

5.1 SPECIAL CASES FOR A SYNTACTIC SUBSET OF LTL

By definition, all transitions of nondeterministic automata have exactly one target state. On the other hand, this number of target states is not restricted in alternating automata, and even empty target state sets are allowed. It is nevertheless easy to modify an alternating automaton into an equivalent nondeterministic one in case the alternating automaton has no transitions with two or more target states: to make the automaton nondeterministic, we can simply replace all transitions having an empty target state set with a transition to a special "sink" state added to the automaton.

Lemma 5.1.1 *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 state not included in Q. Then, 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 such that $\mathcal{L}_{\text{fin}}(\mathcal{A}) = \mathcal{L}_{\text{fin}}(\mathcal{A}')$.*

Proof: Because $|Q'| \leq 1$ for all $\langle q, \Gamma, F, Q' \rangle \in \Delta$, it is easy to see from the definition of $\mathcal{A}'$ that $|Q'| = 1$ for all $\langle q, \Gamma, F, Q' \rangle \in \Delta'$, and thus $\mathcal{A}'$ is a nondeterministic automaton. We show that $\mathcal{A}$ fin-accepts a word $w \in \Sigma^\omega$ iff $\mathcal{A}'$ fin-accepts it.

("$\Rightarrow$") Let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}$ on some $w \in \Sigma^\omega$. 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 $E' \subseteq E$, it follows from the definition that V' is trivially 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 the labelling of such an edge is always consistent. Furthermore, because each node $v' \in V \setminus \{v_0\}$ is a successor of another node $v \in V$, there exists an edge $e \in E$ including v' in its target nodes, and because E' contains all edges of E with a nonempty set of target nodes, it follows that v' is a successor of v also in G'. Finally, because all acceptance sequences of G' are obviously acceptance sequences of the fin-accepting run G, the sequences satisfy the fin-acceptance condition, and it follows that G' is a partial fin-accepting 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 $L(e) = \big\langle L(v), \Gamma, F, \emptyset \big\rangle \in \Delta$ for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$, where $w(i) \in \Gamma$. 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 obviously $\mathcal{L}_{\text{fin}}\big((\mathcal{A}')^{\hat{q}}\big) = \Sigma^\omega$, it follows that $(\mathcal{A}')^{\hat{q}}$ fin-accepts w^{i+1}, and therefore $(\mathcal{A}')^{L'(v)}$ fin-accepts w^i by Lemma 3.4.1. Since this same result holds for all nodes of V' with no outgoing edges, we can apply Proposition 2.3.10 to extend the partial run G' into a fin-accepting run of $\mathcal{A}'$ on w, and thus $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}')$.

("$\Leftarrow$") 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 partial fin-accepting run $G = \langle V, E, L \rangle$ of $\mathcal{A}$ on w as follows:

- $V_0 \overset{\mathit{def}}{=} V_0'$, $V_{i+1} \overset{\mathit{def}}{=} \big\{ v \in V_{i+1}' \,\big|\, L'(v) \neq \hat{q} \big\}$ for all $0 \leq i < \omega$,
- $E \overset{\mathit{def}}{=} \big\{ e \in E' \,\big|\, L'(e) = \langle q, \Gamma, F, Q' \rangle \in \Delta', \hat{q} \notin \{q\} \cup Q' \big\}$, and
- $L(x) \overset{\mathit{def}}{=} L'(x)$ for all $x \in V \cup E$.

Similar to above, because $V \subseteq V'$ and $E \subseteq E'$, V can be partitioned into finite disjoint levels, 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 if such an edge exists, its labelling is consistent.

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 $e' = \langle v, V' \rangle \in E'$ including v' in its target nodes V'. Because $e' \notin E$, however, it follows that $L'(e')$ necessarily includes $\hat{q}$ either as its start state or as one of its target states. By the definition of Δ', $\hat{q}$ is actually the only target state of $L'(e')$, which implies that all nodes of V' are labelled with $\hat{q}$ in G'. Thus especially $L'(v') = \hat{q}$, which however contradicts the fact that $v' \in V$. Therefore v' is necessarily a successor of another node in V.

Because the acceptance sequences of G form a subset of the acceptance sequences of G', all of which satisfy the fin-acceptance condition, it follows that G is a partial fin-accepting 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 $e \in E'$ labelled with a transition $\big\langle L'(v), \Gamma, F, \{\hat{q}\} \big\rangle \in \Delta'$ for some $\Gamma \subseteq \Sigma$ and $F \subseteq \mathcal{F}$ such that $w(i) \in \Gamma$. Because $L'(v) = L(v)$, 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, Lemma 3.4.1 applies trivially to show that $\mathcal{A}^{L(v)}$ fin-accepts w^i. Because v is arbitrary, it follows by Proposition 2.3.10 that G can be extended into a fin-accepting run of $\mathcal{A}$ on w, and therefore $w \in \mathcal{L}_{\text{fin}}(\mathcal{A})$. $\square$

Lemma 5.1.1 gives a way to modify an alternating automaton with no transitions having two or more target states into a nondeterministic automaton that fin-recognizes its language. This leads to the obvious question on whether the result could be combined with the translation of LTL formulas into linear alternating automata to facilitate direct translation of LTL into nondeterministic automata. In other words, we may ask whether there are any LTL formulas that can be translated into linear alternating automata without introducing transitions having two or more target states during the translation. It is easy to see that all propositional formulas have this property: all transitions of automata built using only the basic translation rules (Sect. 3.2) for the atomic formulas and the $\vee$ and $\wedge$ connectives always have an empty set of target states.

We can easily extend this propositional fragment of LTL to a syntactic subclass of temporal formulas that allow a translation into linear alternating automata having no transitions with two or more target states. For this purpose, we first examine the basic translation rules presented in Sect. 3.2 to find conditions under which the linear alternating automata built according to the rules have this property.

Lemma 5.1.2 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two linear alternating automata with no transitions having two or more distinct target states, and let $\mathcal{A}_3$ be an alternating automaton, all transitions of which have an empty set of target states. Then, 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 (where the initial transitions of $\mathcal{A}_3$ determine the initial self-loops of the compound automaton), and*
- *(e) $\mathcal{A}_1$ and $\mathcal{A}_3$ with one of the R translation rules (where the initial transitions of $\mathcal{A}_3$ determine the initial self-loops of the compound automaton)*

are linear alternating automata with no transitions having two or more distinct target states.

Proof: (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 $\mathcal{A}_1$ is a linear alternating automaton, all transitions of which 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 $\mathcal{A}_2$ are linear alternating automata with no transitions having 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, taking the union of the target state sets of any pair of transitions of $\mathcal{A}_1$ and $\mathcal{A}_3$ will result in a set consisting of 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 target state set of some initial transition of $\mathcal{A}_3$; this set is obviously empty for all initial transitions of $\mathcal{A}_3$. Because no transition of $\mathcal{A}_1$ has two or more distinct target states, there can be 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. $\square$

We can now use Lemma 5.1.2 to isolate a simple syntactic fragment of LTL that can be translated into linear alternating automata having no transitions with two or more target states.

Proposition 5.1.3 *Let $\Phi \subseteq LTL(AP)$ be the smallest set of LTL formulas that includes $PL(AP)$ as a subset and is closed under the syntactic rule*

$$\text{If } \varphi_1, \varphi_2 \in \Phi \text{ and } \psi \in PL(AP) \text{, then}$$
$$\mathsf{X}\varphi_1, (\varphi_1 \vee \varphi_2), (\varphi_1 \wedge \psi), (\psi \wedge \varphi_1), (\psi \,\mathsf{U}\, \varphi_1), (\varphi_1 \,\mathsf{R}\, \psi) \in \Phi.$$

Let $\varphi \in \Phi$. The linear alternating automaton built from φ using the translation rules presented in Sect. 3.2 has no transitions with two or more distinct target states.

Proof: It is clear from the basic translation rules that the result holds for all $\varphi \in \Phi$ with $|\varphi| = 1$, i.e., the atomic formulas over AP. Assume that the result holds for all formulas in Φ of size less than or equal to some $1 \leq k < \omega$, and let $\varphi \in \Phi$ be a formula of size $k+1$. Due to the minimality of Φ, φ is a compound formula of one of the above forms, and therefore φ has one or two top-level subformulas in Φ of size at most k. By the induction hypothesis, these subformulas can be translated into linear 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 type of the above compound formulas by Lemma 5.1.2.

For example, if $\varphi = (\psi \,\mathsf{U}\, \varphi_1)$ for some $\psi \in PL(AP)$ and $\varphi_1 \in \Phi$, then we can build linear alternating automata having the desired property for the formulas ψ and φ_1 by the induction hypothesis. More precisely, because ψ is a propositional formula, we can translate ψ into a linear alternating automaton, all transitions of which have an empty set of target states. By Lemma 5.1.2 (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. $\square$

By combining Proposition 5.1.3 and Theorem 3.4.3 with Lemma 5.1.1, we obtain an effective translation procedure from the subset $\Phi \subseteq LTL(AP)$ into nondeterministic automata. By restricting the automata built from formulas in this set using the procedure to the states actually reachable from their initial states, we find that the size of each nondeterministic automaton depends linearly on the number of temporal subformulas in its corresponding formula.

Corollary 5.1.4 *Let $\Phi \subseteq LTL(AP)$ be defined as in Proposition 5.1.3, and let $\varphi \in \Phi$. The language $\mathcal{L}(\varphi)$ can be fin-recognized by a nondeterministic automaton with at most $2 + |\mathrm{Temp}(\varphi)|$ states.*

Proof: Let $\varphi \in \Phi$, let $\mathcal{A}$ be a linear alternating automaton built from φ using the basic translation rules, and let q_I be the initial state of $\mathcal{A}$. By Proposition 5.1.3, $\mathcal{A}$ has no transitions with two or more target states, and thus the subautomaton $\mathcal{A}^{q_I}$ has this same property. We can now apply Lemma 5.1.1 to modify $\mathcal{A}^{q_I}$ into an equivalent nondeterministic automaton by adding one new state to $\mathcal{A}^{q_I}$. The result then follows because $|\mathcal{A}^{q_I}| \leq 1 + |\mathrm{Temp}(\varphi)|$ (Proposition 3.3.1) and because $\mathcal{A}$ and $\mathcal{A}^{q_I}$ fin-accept the same language (Proposition 2.3.11). $\square$

The above results were obtained by examining the original translation rules presented in Sect. 3.2. Obviously, we could further enlarge the set Φ of formulas, for example, by considering also the special cases that permit the use of the improved translation rules presented in Sects. 4.3.2 and 4.3.3. Actually, our running formula translation example illustrates another special case of a formula that can be translated into a linear alternating automaton with no transitions having two or more target states (see Sect. 4.3.4). As a matter of fact, a simple generalization of the example shows that also all formulas of the form $\bigwedge_{i=1}^{n} \mathsf{GF}\psi_i \equiv \bigwedge_{i=1}^{n} \big(\bot \, \mathsf{R}_\mathsf{w} \, (\top \, \mathsf{U}_\mathsf{s} \, \psi_i)\big)$ for some $1 \leq n < \omega$ (where $\psi_i \in PL(AP)$ for all $1 \leq i \leq n$) support translation into such linear alternating automata. However, because of the language containment requirements in the refined translation rules, all special cases of formulas that support similar translation cannot be captured in a general way purely syntactically.

5.2 MEMORYLESS RUNS

In this section, we review a technical result that will allow us to restrict the search for an accepting run for a linear alternating automaton to a subset of runs of the automaton to determine its emptiness. This result leads to a simple general transformation from linear alternating automata to nondeterministic automata.

Apart from the requirement that each level of a run of an alternating automaton must be finite, the general definition of runs of alternating automata does not force any global upper bound for the total number of nodes at a level of the run graph. This flexibility in the definition has helped us to prove several properties of runs: for example, we can always extend a partial accepting run of an alternating automaton into an accepting run by simply "attaching" accepting runs of the automaton's subautomata to each node of the run with no outgoing edges, without any requirements on the structure of the runs of the subautomata (Proposition 2.3.10). However, this flexibility is actually not necessary for capturing acceptance as defined in Sect. 2.3.1.

We shall show that a linear alternating automaton $\mathcal{A}$ accepts a word $w \in \Sigma^\omega$ iff it has an accepting run, each level of which consists of at most $|\mathcal{A}|$ nodes labelled with different states of $\mathcal{A}$. Intuitively, this means that the automaton $\mathcal{A}$ can accept all words in its language without ever spawning more than one copy of a particular subautomaton at each step when working on its input. Of course, which subautomata to spawn at a particular step may still depend nondeterministically on the combinations of transitions that the active copies of $\mathcal{A}$ can take in that step, and not all choices for the subautomata always give rise to an accepting run for the automaton. Nevertheless, the automaton always has at least one "correct" way to choose the subautomata to spawn at each step whenever the input given for the automaton belongs to the language of the automaton; conversely, such a way exists only if the input belongs to this language.

Formally, we call a run $G = \langle V, E, L \rangle$ of an alternating automaton $\mathcal{A}$ *memoryless* iff $|V_i| \leq |\mathcal{A}|$ and $L(v) \neq L(v')$ hold for all $0 \leq i < \omega$ and $v, v' \in V_i$, $v \neq v'$. The terminology reflects the intuition that we do not keep track of the transitions (or transition sequences) that cause a particular subautomaton to be spawned at some step in a run of the automaton. In other words, the behavior of each subautomaton rooted at a given state of the automaton depends only on the level of the run in which it is spawned, not on the automaton's previous actions that caused the subautomaton to be spawned. In the literature, graphs (or trees) representing this kind of behavior of various types of alternating automata have also been called *history-free* or *uniform strategies* by Emerson and Jutla [18] and Muller and Schupp [51, 53], respectively, or in the context of linear alternating automata, *uniform runs* by Rohde [58]. We use the same terminology as Kupferman and Vardi [41].

We now give a formal proof of the above result for our class of linear alternating automata. Actually, Emerson and Jutla [18] have shown the result to hold more generally for all alternating automata (with a *parity* acceptance condition) on infinite trees.

Proposition 5.2.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton simplified in the sense of Corollary 2.3.17. For all $w \in \Sigma^\omega$, $\mathcal{A}$ fin-accepts w iff $\mathcal{A}$ has a memoryless fin-accepting run on w.*

Proof: ("$\Rightarrow$") Let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}$ on w. We use G to find a memoryless fin-accepting run $G' = \langle V', E', L' \rangle$ of $\mathcal{A}$ on w by choosing, for each level of G, a set of transitions labelling a subset of edges starting from the level and then forming the following level of G' by choosing a representative node for each distinct state of $\mathcal{A}$ labelling a target node of one of these edges. These representatives then guide the selection of another set of transitions.

Clearly, forming each level of G' from nodes labelled with distinct states of $\mathcal{A}$ already ensures that each level of G' will consist of at most $|\mathcal{A}|$ nodes, none of which shares its label with another node in the same level. However, the requirement that G' should be a fin-accepting run of $\mathcal{A}$ forces additional constraints on the choice of transitions described above. Namely, we have to ensure that no infinite collection of successive levels of G' defined using sets of transitions includes a (fragment of an) acceptance sequence that violates the fin-acceptance condition. More precisely, because $\mathcal{A}$ is assumed to be simplified in the sense of Corollary 2.3.17, these levels must not include an infinite sequence of consecutive edges labelled with self-loops of $\mathcal{A}$ sharing an acceptance condition in $\mathcal{F}$.

We now give the formal inductive definition of G'. Let $V_0' \overset{def}{=} \{v_0'\}$, and let $L'(v_0') \overset{def}{=} q_I$. To express the above requirement concerning acceptance conditions, we assume the existence of a total order $\prec$ on the finite set of acceptance conditions $\mathcal{F}$. For each level $0 \leq i < \omega$ of G', we shall then define a function $\widetilde{F}_i : Q \to 2^{\mathcal{F}}$; intuitively, an acceptance condition $f \in \mathcal{F}$ will belong to $\widetilde{F}_i(q)$ only if the preceding levels of G' include a nontrivial path labelled with self-loops of $\mathcal{A}$ having q as their start state and including f in their acceptance conditions. Since the first level V_0' of G' has no preceding levels, we let $\widetilde{F}(q) \overset{def}{=} \emptyset$ 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 collecting all transitions of $\mathcal{A}$ occurring as labels of an edge starting from some i^{th}-level node of G labelled with q; formally,

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

Assume that the labels of the nodes in the i^{th} level of G' form a subset of the node labels in the corresponding level of G, i.e., $L'(V_i') \subseteq L(V_i)$; this clearly holds for the level V_0' of G'. Because G is a run of $\mathcal{A}$, this implies that $T_i(q) \neq \emptyset$ for all $q \in L'(V_i')$. We now choose for each such q a transition $t_{i,q} = \langle q, \Gamma_{i,q}, F_{i,q}, Q_{i,q}\rangle \in T_i(q)$ such that $\min_{\prec} \widetilde{F}_i(q) \notin F_{i,q}$ if $\widetilde{F}_i(q) \neq \emptyset$ and such a transition exists in $T_i(q)$; otherwise we let $t_{i,q}$ be any transition in $T_i(q)$. Thus, if the preceding levels of G' include a path corresponding to a sequence of self-loops sharing a common acceptance condition, we always try to extend this sequence with a transition that shares fewer acceptance conditions with the transitions in the sequence if possible. We then collect all target states of the transitions $t_{i,q}$ into a set $Q_{i+1} \overset{def}{=} \bigcup_{q \in L'(V_i')} Q_{i,q} = \{q_{i,1}, \ldots, q_{i,n_i}\}$ (where $0 \leq n_i < \omega$, and $q_{i,j} \neq q_{i,k}$ for all $1 \leq j, k \leq n_i$, $j \neq k$) and define the $(i+1)^{\text{th}}$ level of G' by taking a representative node for each state $q' \in Q_{i+1}$, i.e.,

$$V_{i+1}' \overset{def}{=} \{v_{i,1}, \ldots, v_{i,n_i}\} \quad \text{and} \quad L'(v_{i,j}) \overset{def}{=} q_{i,j} \text{ for all } 1 \leq j \leq n_i.$$

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

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

and $\widetilde{F}_{i+1}(q) \stackrel{def}{=} \emptyset$ otherwise. In the first case, because $q \in Q_{i,q}$, the transition $t_{i,q}$ is a self-loop of $\mathcal{A}$ starting from the state q. Because $\widetilde{F}_i(q) = \emptyset$, however, the preceding levels of G' do not contain a nontrivial path of self-loops (starting from the state q) sharing a common acceptance condition. Therefore, $t_{i,q}$ is the first transition in such a sequence of self-loops through q, and thus we collect all acceptance conditions of $t_{i,q}$ into $\widetilde{F}_{i+1}(q)$.

If $\widetilde{F}_i(q) \neq \emptyset$, then the preceding levels of G' contain a nontrivial path of self-loops sharing a common acceptance condition. The second case of the definition now guarantees that this condition will still be included in $\widetilde{F}_{i+1}(q)$ iff also $t_{i,q}$ is such a self-loop. Therefore, $\widetilde{F}_{i+1}(q)$ still has the intended meaning in the $(i+1)^{\text{th}}$ level of G'. This completes the inductive definition of the functions $\widetilde{F}_i$.

Finally, 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 if $e = \langle v, V'' \rangle \in E'$, then $L'(e) \stackrel{def}{=} t_{i,L'(v)}$.

We check that G' is a memoryless run of $\mathcal{A}$ on w. It follows directly from the definitions of V', E' and L' that V'_0 consists of a single node labelled with the initial state of $\mathcal{A}$, V' is partitioned into finite disjoint levels that contain at most $|\mathcal{A}|$ nodes, the nodes at each level of G' have different labels, and the edges of E' lie between successive levels of G'.

Let $v \in V'_i$ for some $0 \le i < \omega$, and let $L'(v) = q$. It is clear from the above construction that G contains at least one i^{th}-level node labelled with the state q, and because all nodes of G have a unique outgoing edge, it follows that $T_i(q) \neq \emptyset$. Therefore the transition $t_{i,q} = \langle q, \Gamma_{i,q}, F_{i,q}, Q_{i,q} \rangle \in T_i(q)$ labelling one of these edges $e \in E$ in G is well-defined. By the definition of G', $Q_{i,q} \subseteq L'(V'_{i+1})$, 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 in the $(i+1)^{\text{th}}$ level of G' are labelled with distinct states of $\mathcal{A}$. Because $L'(e') = t_{i,q} = L(e)$, it follows that the edge labelling L' is consistent.

If $v' \in V'_i$ for some $1 \le i < \omega$, then there exists a transition $t_{i-1,q} = \langle q, \Gamma_{i-1,q}, F_{i-1,q}, Q_{i-1,q} \rangle \in T_{i-1}(q)$ such that $L'(v') \in Q_{i-1,q}$. 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 $L'(V'') = Q_{i-1,q}$. 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'. Therefore, G' is a run of $\mathcal{A}$ on w.

Assume that G' is not a fin-accepting run of $\mathcal{A}$. Therefore G' contains an acceptance sequence $(e_i)_{0 \le i < \omega} \in \mathcal{E}(G')$ that does not satisfy the fin-acceptance condition. There now exists a minimal index $0 \le j < \omega$ and a maximal nonempty subset $F \subseteq \mathcal{F}$ such that for all $j \le i < \omega$, the transition $L'(e_i)$ includes all elements of F in its set of acceptance conditions.

Therefore, $L'(e_i) = t_{i,q} = \langle q, \Gamma_{i,q}, F_{i,q}, Q_{i,q}\rangle$ is a self-loop of $\mathcal{A}$ starting from some fixed state $q \in Q$ for all $j \leq i < \omega$, because the set of acceptance conditions associated with every non-self-loop transition of $\mathcal{A}$ is empty ($\mathcal{A}$ is simplified in the sense of Corollary 2.3.17).

If $\widetilde{F}_i(q) \neq \emptyset$ for all $j \leq i < \omega$, then, by the definition of the functions $\widetilde{F}_i$, there necessarily exists an acceptance condition $f \in \bigcap_{j\leq i<\omega} \widetilde{F}_i(q)$, and thus $f \in \bigcap_{j-1\leq i<\omega} F_{i,q}$. However, this contradicts the minimality of j. Thus there exists an index $j \leq k < \omega$ such that $\widetilde{F}_k(q) = \emptyset$.

Because $\widetilde{F}_k(q) = \emptyset$ and $t_{k,q}$ is a self-loop of $\mathcal{A}$, $\widetilde{F}_{k+1}(q) = F_{k,q}$. Thus $F \subseteq \widetilde{F}_{k+1}(q)$, because $F \subseteq F_{k,q}$ holds by assumption. Then, $F \subseteq \widetilde{F}_{k+i}(q)$ holds for all $1 \leq i < \omega$, because $F \subseteq \widetilde{F}_{k+i}(q)$ for some $1 \leq i < \omega$ implies that $\widetilde{F}_{k+i+1}(q) = F_{k+i,q} \cap \widetilde{F}_{k+i}(q) \supseteq F$ ($F \subseteq F_{k+i,q}$ holds by assumption). Furthermore, because the sizes of the sets $\widetilde{F}_i(q)$ ($k \leq i < \omega$) form a non-increasing sequence of positive integers, it follows that there exists an index $k \leq l < \omega$ such that $\widetilde{F}_l(q) = \widetilde{F}_{l+i}(q)$ for all $0 \leq i < \omega$. Thus there exists an acceptance condition $f \in F$ such that $f = \min_{\prec} \widetilde{F}_i(q)$ for all $l \leq i < \omega$.

Let $l \leq i < \omega$, and let $v \in V_i$ be a node in G labelled with q (such a node always exists, because $L'(V_i') \subseteq L(V_i)$, and V_i' always contains the start node of the edge e_i, labelled with q by assumption). Because $f = \min_{\prec} \widetilde{F}_i(q) \in F_{i,q}$, it now follows that f belongs to the acceptance conditions of the transition labelling the edge starting from v in G, since otherwise we could have defined $t_{i,q}$ to be this transition (and thus $f \notin F_{i,q}$). Because only self-loops of $\mathcal{A}$ can have a nonempty set of acceptance conditions, this transition is a self-loop, and thus v has a successor in G also labelled with q, because the labelling L is consistent. By induction on i, it now follows that G contains an acceptance sequence having an infinite suffix of edges labelled with self-loops of $\mathcal{A}$ sharing the acceptance condition f, which contradicts the assumption that G is a fin-accepting run of $\mathcal{A}$ on w. Therefore, no acceptance sequence of G' can violate the fin-acceptance condition, and G' is a memoryless fin-accepting run of $\mathcal{A}$ on w as argued.

("$\Leftarrow$") This result follows immediately from the fact that all memoryless runs of $\mathcal{A}$ on w are runs of $\mathcal{A}$ on w. Thus, if a memoryless run of $\mathcal{A}$ on w is fin-accepting, then $\mathcal{A}$ fin-accepts w. □

5.3 GENERAL TRANSLATION

As shown by Miyano and Hayashi [49], any alternating automaton with n states (working on infinite words in inf-acceptance mode using a single acceptance condition associated with the states of the automaton) can be transformed into an equivalent nondeterministic automaton with at most 4^n states and a single acceptance condition. The construction of Isli [37], which can actually be seen as a slight rearrangement of Miyano and Hayashi's construction, allows the upper bound to be reduced to $(\frac{2}{3})^m 3^n$ states, where m is the number of states associated with the single acceptance condition of the automaton. Muller, Saoudi and Schupp presented a $n4^n$ construction for weak alternating automata on infinite trees [51]. Linear automata on words were further studied by Rohde [58], who gave a construction for automata

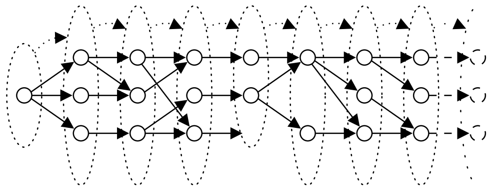


Fig. 5.1: Collapsing the levels of a run of an alternating automaton into a nonbranching sequence of nodes

on transfinite words, and Gastin and Oddoux [28], who gave a 2^n upper bound for the translation of linear alternating automata (still, with a single acceptance condition associated with the states of the automaton) into nondeterministic automata with at most m generalized acceptance conditions on transitions. In this section we generalize their result to linear alternating automata, where we already allow the alternating automata to have multiple acceptance conditions on their transitions. While the corresponding nondeterministic automaton will still have at most 2^n states, associating the acceptance conditions with the transitions of the alternating automaton will force the nondeterministic automaton to have a total of nm acceptance conditions in the worst case (where m denotes the number of generalized acceptance conditions in the alternating automaton).

Nondeterminization Construction

Consider a memoryless run of a linear alternating automaton. Because each level of this run comprises a (possibly empty) set of nodes labelled with distinct states of the automaton, the labels of the nodes in the level form a finite subset of states of the automaton. Intuitively, we can identify this run with a run of another automaton on the same input by collapsing each individual level of the run into a single node and the edges between each pair of consecutive levels into a single edge between the nodes corresponding to the levels as shown in Fig. 5.1. We thus obtain a nonbranching node sequence, which we then identify with a run of another automaton by imposing a labelling (defined below) on the nodes and edges in this sequence. Because each node in this run has only one successor, each transition labelling an edge in the run cannot have two or more distinct target states in order for the labelling to be consistent. Therefore, because each run of the linear alternating automaton can be identified with a similar nonbranching node sequence, it follows that the underlying automaton can actually be made nondeterministic. Formally, we build the states of this automaton from subsets of states of the linear alternating automaton, and the transitions of the nondeterministic automaton are obtained by "synchronizing" sets of transitions starting from a given subset of states.

Another way to view the construction is to compare it to the basic *subset construction* used for determinizing nondeterministic automata on finite inputs. In our case, however, the subsets consist of the current states of the active copies of the alternating automaton (operating in parallel on the input), instead of representing the set of possible current states for a single copy of the automaton. This difference is reflected also in the definition of the transition relation, which is formed by merging several transitions taken synchronously by the active copies of the alternating automaton into a single transition.

Theorem 5.3.1 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton simplified in the sense of Corollary 2.3.17. Define the automaton $\mathcal{A}' = \big\langle \Sigma, 2^Q, \Delta', \{q_I\}, Q \times \mathcal{F} \big\rangle$, where the transition relation Δ' satisfies, for all $Q' = \{q_1, \ldots, q_n\} \in 2^Q$ for some $0 \leq n < \omega$, $\Gamma \subseteq \Sigma$, $F \subseteq Q \times \mathcal{F}$ and $\mathcal{Q} \subseteq 2^Q$,*

$$\big\langle Q', \Gamma, F, \mathcal{Q} \big\rangle \in \Delta' \quad \text{iff} \quad \text{there exist transitions } \langle q_i, \Gamma_i, F_i, Q'_i \rangle \in \Delta \ (1 \leq i \leq n) \text{ such that } \Gamma = \bigcap_{1 \leq i \leq n} \Gamma_i,\ F = \bigcup_{1 \leq i \leq n} \big(\{q_i\} \times F_i\big), \text{ and } \mathcal{Q} = \big\{ \bigcup_{1 \leq i \leq n} Q'_i \big\}.$$

Then, $\mathcal{A}'$ is a nondeterministic automaton for which $\mathcal{L}_{\text{fin}}(\mathcal{A}') = \mathcal{L}_{\text{fin}}(\mathcal{A})$.

Proof: It is clear from the definition that the target states of each transition of $\mathcal{A}'$ always form a singleton set (containing a subset of Q), and thus $\mathcal{A}'$ is nondeterministic. We show that $\mathcal{A}$ and $\mathcal{A}'$ fin-accept the same language.

("$\Rightarrow$") Let $G = \langle V, E, L \rangle$ be a fin-accepting run of $\mathcal{A}$ on $w \in \Sigma^\omega$. By Proposition 5.2.1, we may assume that G is memoryless. Define the graph $G' = \langle V', E', L' \rangle$, where $V' \overset{\text{def}}{=} \bigcup_{0 \leq i < \omega} \{v'_i\}$ (with $V'_i \overset{\text{def}}{=} \{v'_i\}$ for all $0 \leq i < \omega$), $E' \overset{\text{def}}{=} \bigcup_{0 \leq i < \omega} \big\{ \langle v'_i, V'_{i+1} \rangle \big\}$, and $L'(v'_i) \overset{\text{def}}{=} L(V_i)$ for all $0 \leq i < \omega$. Let $0 \leq i < \omega$; to define the label of the edge starting from the node v'_i, we first denote $V_i = \{v_{i,1}, \ldots, v_{i,n_i}\}$ for some $0 \leq n_i < \omega$ ($v_{i,j} \neq v_{i,k}$ for all $1 \leq j, k \leq n$, $j \neq k$). We then collect the transitions $T_i \subseteq \Delta$ labelling the edges starting from nodes at level i of G:

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

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

We check that G' is a run of $\mathcal{A}'$ on w. First, we see that $V'_0 = \{v'_0\}$ and $L'(v'_0) = L(V_0) = L\big(\{v_0\}\big) = \{q_I\}$ (the initial state of $\mathcal{A}'$), and G' is partitioned into finite disjoint levels with edges between consecutive levels.

Let $v'_i \in V$ for some $0 \leq i < \omega$. By the definition of G', v'_i has the unique outgoing edge $\langle v'_i, V'_{i+1} \rangle \in E'$. Because G is memoryless, it follows that $L(v_{i,j}) \neq L(v_{i,k})$ for all $1 \leq j, k \leq n_i$, $j \neq k$, and thus T_i consists of n_i transitions with distinct start states. From the definition of $\mathcal{A}'$ it now follows that Δ' contains the transition $t = \big\langle \bigcup_{1 \leq j \leq n_j} \{q_{i,j}\}, \bigcap_{1 \leq j \leq n_i} \Gamma_{i,j}, \bigcup_{1 \leq j \leq n_i} \big(\{q_{i,j}\} \times F_{i,j}\big), \big\{ \bigcup_{1 \leq j \leq n_i} Q'_{i,j} \big\} \big\rangle$. Because T_i consists of the labels of the edges leaving the nodes at level i of G, $\bigcup_{1 \leq j \leq n_j} \{q_{i,j}\} = L(V_i) = L'(v'_i)$; furthermore, the fact that G is a run implies that $w(i) \in \Gamma_{i,j}$ for all $1 \leq j \leq n_i$, and $\bigcup_{1 \leq j \leq n_i} Q'_{i,j} = L(V_{i+1})$. Therefore $w(i) \in \bigcap_{1 \leq j \leq n_i} \Gamma_{i,j}$ and $\big\{ \bigcup_{1 \leq j \leq n_i} Q'_{i,j} \big\} = \big\{ L(V_{i+1}) \big\} = \big\{ L'(v'_{i+1}) \big\} = L'(V'_{i+1})$, and it follows that t is equal to the label of the edge $\langle v'_i, V'_{i+1} \rangle$, and the labelling L' is consistent.

Finally, it is easy to see from the definition of E' that every node $v' \in V'_i$ ($1 \leq i < \omega$) is a successor of another node in V'_{i-1}, and therefore G' is a run of $\mathcal{A}'$ on w.

Assume that G' is not a fin-accepting run of $\mathcal{A}'$. Then there exists a $0 \leq j < \omega$ and an acceptance condition $\langle q, f\rangle \in Q \times \mathcal{F}$ such that the transition $L'\big(\langle v'_i, V'_{i+1}\rangle\big)$ includes $\langle q, f\rangle$ in its acceptance conditions for all $j \leq i < \omega$. By the definition of L', this implies that for all $j \leq i < \omega$, T_i contains a transition $t_i \in \Delta$ with start state q such that t_i includes f in its acceptance conditions. Because $\mathcal{A}$ is simplified in the sense of Corollary 2.3.17, all transitions t_i are self-loops of $\mathcal{A}$. Therefore, because each t_i labels an edge starting from a node $v_i \in V_i$ with $L(v_i) = q$, v_i has also a successor in V_{i+1} labelled with the state q, because the labelling L is consistent. Because G is a memoryless run of $\mathcal{A}$, no two nodes of V_{i+1} have the same label, and therefore v_i has v_{i+1} as its successor. But then, by induction, G contains an acceptance sequence ending in an infinite suffix of edges corresponding to transitions, all of which include f in their acceptance conditions. This contradicts the assumption that G is a fin-accepting run of $\mathcal{A}$. Therefore G' is necessarily a fin-accepting run of $\mathcal{A}'$ on w, and $\mathcal{L}_{\text{fin}}(\mathcal{A}) \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A}')$.

("$\Leftarrow$") Let $G' = \langle V', E', L'\rangle$ be a fin-accepting run of $\mathcal{A}'$ on $w \in \Sigma^\omega$. Without loss of generality, we may assume that G' is memoryless; because $\mathcal{A}'$ is nondeterministic, each level V'_i of G' consists of a single node v'_i for all $0 \leq i < \omega$. Let $L'(v'_i) = \{q_{i,1}, \ldots, q_{i,n_i}\} \in 2^Q$ for some $0 \leq n_i < \omega$ (where $q_{i,j} \neq q_{i,k}$ for all $1 \leq j, k \leq n_i$, $j \neq k$) for all $0 \leq i < \omega$.

We define a run $G = \langle V, E, L\rangle$ of $\mathcal{A}$ on w. For all $0 \leq i < \omega$, let V_i consist of n_i new nodes $v_{i,1}, \ldots, v_{i,n_i}$, and let $L(v_{i,j}) \stackrel{def}{=} q_{i,j}$ 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 memoryless run of a nondeterministic automaton, the node $v'_i \in V'$ has the unique outgoing edge $\big\langle v'_i, \{v'_{i+1}\}\big\rangle \in E'$ labelled with a transition $\big\langle L'(v'_i), \Gamma_i, F_i, \{L'(v'_{i+1})\}\big\rangle \in \Delta'$ 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 now 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 $\Gamma_i = \bigcap_{1 \leq j \leq n_i} \Gamma_{i,j}$, $F_i = \bigcup_{1 \leq j \leq n_i} \big(\{q_{i,j}\} \times F_{i,j}\big)$ and $\{L'(v'_{i+1})\} = \big\{\bigcup_{1 \leq j \leq n_i} Q'_{i,j}\big\}$.

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

We check that G is a run of $\mathcal{A}$ on w. Clearly, because $L'(v'_0) = \{q_I\} = L(V_0)$, it follows that V_0 consists of a single node labelled with q_I. Because the nodes of G' are labelled with finite subsets of Q, it follows that V_i is finite for all $0 \leq i < \omega$, and the levels of G are disjoint by the definition. By the definition of E, there are edges only between consecutive levels of G.

Let $v \in V_i$ for some $0 \leq i < \omega$. Then $v = v_{i,j}$ for some $1 \leq j \leq n_i$, and v has the unique outgoing edge $e_{i,j} = \langle v, V''\rangle \in E$ labelled with the transition $t_{i,j} = \langle q_{i,j}, \Gamma_{i,j}, F_{i,j}, Q'_{i,j}\rangle \in \Delta$. Clearly, $L(v) = q_{i,j}$, and from the definition of $e_{i,j}$ we see that $L(V'') = Q'_{i,j}$. Furthermore, because G' is a run, $w(i) \in \Gamma_i = \bigcap_{1 \leq k \leq n_i} \Gamma_{i,k}$. Thus especially $w(i) \in \Gamma_{i,j}$, and the labelling L is consistent.

Let $v \in V_i$ for some $1 \leq i < \omega$. Then $L(v) = q \in L'(v'_i)$. Because

G' is a memoryless run of a nondeterministic automaton, v'_i is a successor of the node $v'_{i-1} \in V'$. Because the edge between these nodes is labelled with a transition of $\mathcal{A}'$ and 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 $t_{i-1,j} \in \Delta$ starting from a state $q_{i-1,j} \in L'(v'_{i-1})$ that includes q in its target states. By the definition of G, V_{i-1} now contains a node $v_{i-1,j}$ with $L(v_{i-1,j}) = q_{i-1,j}$, and the edge starting from this node is labelled with the transition $t_{i-1,j}$. Thus $v_{i-1,j}$ has a successor labelled with the state q. This successor is now necessarily the node v, because v is the unique node in V_i labelled with the state q. It follows that G is a run of $\mathcal{A}$ on w.

Assume 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 $\mathcal{E}(G)$ contains an acceptance sequence $(e_i)_{0 \leq i < \omega}$, where the transition $L(e_i)$ includes f in its acceptance conditions for all $j \leq i < \omega$. By Proposition 2.3.15, we may choose j such that the transitions $L(e_i)$ have the same start state $q \in Q$ for all $j \leq i < \omega$. Because $e_i \in E$ for all $j \leq i < \omega$, then $L(e_i) = t_{i,k} = \langle q, \Gamma_{i,k}, F_{i,k}, Q'_{i,k} \rangle$ for some $1 \leq k \leq n_i$. But then, because $f \in F_{i,k}$, it follows that $\langle q, f \rangle \in F_i$ for all $j \leq i < \omega$. Thus the transition $L'\big(\langle v'_i, \{v'_{i+1}\}\rangle\big)$ includes the pair $\langle q, f \rangle$ in its acceptance conditions 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 $\mathcal{L}_{\text{fin}}(\mathcal{A}') \subseteq \mathcal{L}_{\text{fin}}(\mathcal{A})$. □

Number of Acceptance Conditions: A Worst Case Example

Translating a linear alternating automaton into an equivalent nondeterministic automaton using the construction in Theorem 5.3.1 results in a blow-up in the number of acceptance conditions that is linear in the number of states of the automaton. An obvious question is whether this blow-up is actually necessary. In the general case, the blow-up proves to be unavoidable if we wish to use the same construction for defining the states and transitions of the nondeterministic automaton. We illustrate this fact with a simple example.

Example 5.3.2 Let $\mathcal{A}$ be the linear alternating automaton shown in Fig. 5.2 (a). Figure 5.2 (b) shows the transition structure of the nondeterministic automaton obtained from $\mathcal{A}$ using the construction given in Theorem 5.3.1 and then removing all states not reachable from the initial state of the resulting automaton. Instead of defining the acceptance conditions of this automaton as specified in the construction (in which case the nondeterministic automaton would have two distinct acceptance conditions), we check whether it is possible to make the automaton fin-accept $\mathcal{A}$'s language with only one acceptance condition.

Clearly, if the transition t_1 has no acceptance conditions, then the nondeterministic automaton fin-accepts the word a^ω, which does not, however, belong to $\mathcal{L}_{\text{fin}}(\mathcal{A})$. On the other hand, leaving the transition t_2 with an empty set of acceptance conditions would lead to the acceptance of the word ab^ω, which is not fin-accepted by $\mathcal{A}$, either. Therefore we need to associate both t_1 and t_2 with an acceptance condition. However, if we give the same condition for both transitions, the nondeterministic automaton will no longer fin-accept the word $(ab)^\omega \in \mathcal{L}_{\text{fin}}(\mathcal{A})$. It follows that we need at least two acceptance conditions to make the languages accepted by the two automata

Fig. 5.2: (a) The linear alternating automaton $\mathcal{A} = \big\langle\{a,b\}, \{q_1,q_2\}, \big\{\langle q_1, \{a\}, \{f\}, \{q_1,q_2\}\rangle, \langle q_1, \{b\}, \emptyset, \{q_1\}\rangle, \langle q_2, \{a\}, \emptyset, \{q_2\}\rangle, \langle q_2, \{b\}, \{f\}, \{q_2\}\rangle\big\}, q_1, \{f\}\big\rangle$;
(b) Transition structure of the nondeterministic automaton obtained from $\mathcal{A}$ by applying the construction in Theorem 5.3.1

equivalent without modifying the structure of the nondeterministic automaton. By Theorem 5.3.1, two acceptance conditions are also sufficient in this case. ■

It is easy to see from the definition of the translation in Theorem 5.3.1 that the nondeterministic automaton will include a transition associated with an acceptance condition $\langle q, f\rangle \in Q \times \mathcal{F}$ iff f is included in the acceptance conditions of some transition that starts from the state q in the linear alternating automaton. Because the acceptance conditions not associated with any transition do not affect fin-acceptance (clearly, no such condition can cause any acceptance sequence to violate the fin-acceptance condition in a run of the automaton), we can remove these conditions from the automaton. The upper bound for the number of acceptance conditions in the nondeterministic automaton can thus be improved to

$$\sum_{f\in\mathcal{F}} \Big|\big\{q \in Q \,\big|\, f \in F \text{ for some } \langle q, \Gamma, F, Q'\rangle \in \Delta\big\}\Big| \leq |Q| \times |\mathcal{F}|.$$

Size of a Nondeterministic Automaton Corresponding to an LTL Formula
Theorem 5.3.1 leads to the following corollary regarding the complexity of translation of linear temporal logic formulas into nondeterministic automata. This result is essentially equivalent to the one given by Couvreur [14] and Gastin and Oddoux [28] with the exception of an additive constant (that arises as a consequence of restricting to automata with only a single initial state).

Corollary 5.3.3 *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 with at most $1 + 2^{|\mathrm{Temp}(\varphi)|}$ states. (If φ is a binary pure temporal formula, the upper bound reduces to $2^{|\mathrm{Temp}(\varphi)|}$.)*

Proof: Let φ be an LTL formula in the given form. By Corollary 3.3.2, we can apply the basic translation rules to find a linear alternating automaton $\mathcal{A}$ having at most $1 + |\mathrm{Temp}(\varphi)|$ states ($|\mathrm{Temp}(\varphi)|$ states if φ is a binary pure temporal formula) such that $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}) = \mathcal{L}(\varphi)$. By the construction in Theorem 5.3.1, there exists a nondeterministic automaton $\mathcal{A}'$ that fin-accepts the same language. Because this automaton consists of $2^{|\mathcal{A}|}$ states, the result now follows immediately if φ is a binary pure temporal formula; otherwise the construction in Theorem 5.3.1 yields a nondeterministic automaton with at most $2^{1+|\mathrm{Temp}(\varphi)|}$ states. In this case, however, because φ is not a binary pure temporal formula, we see from the basic translation rules that the automaton $\mathcal{A}$ has no self-loop transitions starting from its initial state q_I; furthermore, because $\mathcal{A}$ is a linear alternating automaton, no other transition of $\mathcal{A}$ can have q_I as its target state, either. By the construction of $\mathcal{A}'$, this allows us to remove from $\mathcal{A}'$ all states that contain the state q_I except for the initial state $\{q_I\}$ of $\mathcal{A}'$, since none of these states are reachable from $\{q_I\}$ in $\mathcal{A}'$. Therefore, the upper bound for the size of the automaton reduces to $1 + 2^{|\mathrm{Temp}(\varphi)|}$ states, and the result follows. $\square$

Languages Accepted by Subautomata of the Nondeterministic Automaton
We end this section with another corollary of Theorem 5.3.1 that extends the result into subautomata of the nondeterministic automaton obtained from a linear alternating automaton $\mathcal{A}$ using the construction. Each of these subautomata accepts precisely the intersection of the languages accepted by the collection of $\mathcal{A}$'s subautomata, whose initial states comprise the initial state of the nondeterministic subautomaton.

Corollary 5.3.4 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a linear alternating automaton, and let $\mathcal{A}' = \big\langle \Sigma, 2^Q, \Delta', \{q_I\}, Q \times \mathcal{F} \big\rangle$ be the nondeterministic automaton obtained from $\mathcal{A}$ by applying the construction in Theorem 5.3.1. Let $Q' \subseteq Q$. Then, $\bigcap_{q \in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \mathcal{L}_{\mathrm{fin}}\big((\mathcal{A}')^{Q'}\big)$.*

Proof: Let $Q' = \{q_1, \ldots, q_n\}$ for some $0 \le n < \omega$. We first modify $\mathcal{A}$ into another linear alternating automaton $\mathcal{A}^+ = \langle \Sigma, Q^+, \Delta^+, \hat{q}, \mathcal{F} \rangle$, where $Q^+ \stackrel{def}{=} Q \cup \{\hat{q}\}$ for some new state $\hat{q}$ not included in Q, and

$$\Delta^+ \stackrel{def}{=} \Delta \cup \Big\{ \langle \hat{q}, \Gamma, \emptyset, Q'' \rangle \,\Big|\, \text{there exist transitions } \langle q_i, \Gamma_i, F_i, Q'_i \rangle \in \Delta \ (1 \le i \le n) \text{ such that } \Gamma = \textstyle\bigcap_{1 \le i \le n} \Gamma_i \text{ and } Q'' = \bigcup_{1 \le i \le n} Q'_i \Big\}.$$

Clearly, Δ^+ contains no transitions including $\hat{q}$ in their target states, and $(\mathcal{A}^+)^q = \mathcal{A}^q$ holds for all $q \in Q$.

By Theorem 5.3.1, there now exists a nondeterministic automaton $\mathcal{A}'' = \big\langle \Sigma, 2^{Q^+}, \Delta'', \{\hat{q}\}, Q^+ \times \mathcal{F} \big\rangle$ (where Δ'' is defined as described in the theorem) such that $\mathcal{L}_{\mathrm{fin}}(\mathcal{A}'') = \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^+)$. It is easy to see that no target state of a transition of $\mathcal{A}''$ corresponds to a subset of Q^+ that contains the state $\hat{q}$, and thus each target state of a transition of $\mathcal{A}''$ is a subset of Q. Furthermore, $(\mathcal{A}'')^{Q''} = (\mathcal{A}')^{Q''}$ holds for all $Q'' \subseteq Q$.

Now,

$w \in \bigcap_{q \in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q)$

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

iff $\mathcal{A}^q$ fin-accepts w for all $q \in Q' = \{q_1, \ldots, q_n\}$ *(def. of acceptance)*

iff for all $1 \le i \le n$, there exists a transition $\langle q_i, \Gamma_i, F_i, Q'_i \rangle \in \Delta$ such that $w(0) \in \Gamma_i$ and $(\mathcal{A}^{q_i})^{q'}$ $(= \mathcal{A}^{q'})$ fin-accepts w^1 for all $q' \in Q'_i$ *(Lemma 3.4.1)* iff there exists a transition $\langle \hat{q}, \bigcap_{1\le i\le n} \Gamma_i, \emptyset, \bigcup_{1\le i\le n} Q'_i \rangle \in \Delta^+$ such that $w(0) \in \bigcap_{1\le i\le n} \Gamma_i$, and $\mathcal{A}^{q'}$ $(= (\mathcal{A}^+)^{q'})$ fin-accepts w^1 for all $q' \in \bigcup_{1\le i\le n} Q'_i$ *(definition of $\mathcal{A}^+$)*

iff $\mathcal{A}^+$ fin-accepts w *(Lemma 3.4.1)*

iff $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}^+)$ *(definition of acceptance)*

iff $w \in \mathcal{L}_{\text{fin}}(\mathcal{A}'')$ *(Theorem 5.3.1)*

iff $\mathcal{A}''$ fin-accepts w *(definition of acceptance)*

iff there exists a transition $\langle \{\hat{q}\}, \Gamma, \{\hat{q}\} \times F, \{Q''\} \rangle \in \Delta''$ for some $\Gamma \subseteq \Sigma$, $F \subseteq \mathcal{F}$ and $Q'' \subseteq Q$ such that $w(0) \in \Gamma$, and $(\mathcal{A}'')^{Q''}$ fin-accepts w^1 *(definition of $\mathcal{A}''$ and Lemma 3.4.1)*

iff there exists a transition $\langle \hat{q}, \Gamma, F, Q'' \rangle \in \Delta^+$ such that $w(0) \in \Gamma$, and $(\mathcal{A}'')^{Q''}$ fin-accepts w^1 *(definition of $\mathcal{A}''$)*

iff there exist transitions $\langle q_i, \Gamma'_i, F'_i, Q''_i \rangle \in \Delta$ $(1 \le i \le n)$ such that $w(0) \in \Gamma = \bigcap_{1\le i\le n} \Gamma'_i$ and $Q'' = \bigcup_{1\le i\le n} Q''_i$, and $(\mathcal{A}'')^{Q''}$ fin-accepts w^1 *(definition of $\mathcal{A}^+$)*

iff there exists a transition $\langle Q', \Gamma, \bigcup_{1\le i\le n}(\{q_i\} \times F'_i), \{Q''\} \rangle \in \Delta'$ such that $w(0) \in \Gamma$, and $(\mathcal{A}'')^{Q''}$ fin-accepts w^1 *(definition of $\mathcal{A}'$)*

iff there exists a transition $\langle Q', \Gamma, \bigcup_{1\le i\le n}(\{q_i\} \times F'_i), \{Q''\} \rangle \in \Delta'$ such that $w(0) \in \Gamma$, and $(\mathcal{A}')^{Q''}$ fin-accepts w^1 *($(\mathcal{A}'')^{Q''} = (\mathcal{A}')^{Q''}$)*

iff $(\mathcal{A}')^{Q'}$ fin-accepts w *(Lemma 3.4.1)*

iff $w \in \mathcal{L}_{\text{fin}}\big((\mathcal{A}')^{Q'}\big)$. *(definition of acceptance)*

$\square$

Corollary 5.3.4 can be used to reduce the problem of checking the emptiness of the intersection of languages accepted by subautomata of a given linear alternating automaton $\mathcal{A}$ to checking the emptiness of a subautomaton of the nondeterministic automaton obtained from $\mathcal{A}$ using the construction presented in Theorem 5.3.1. This provides us with a method for handling language emptiness checks such as those that arose during our investigation of methods for transition redundancy analysis in Sect. 4.4.

Improvements to the Nondeterminization Construction

In practice, the standard way to construct a nondeterministic automaton equivalent to a linear alternating automaton (or the intersection of its subautomata as discussed above) is to build only the part of the automaton, the state set of which includes the initial state of the automaton and is closed under the automaton's transition relation Δ'. Although not discussed here in detail, this phase should be combined with additional optimizations for on-the-fly removal of transitions from the nondeterministic automaton during the construction, for example, by applying the techniques presented by Gastin and Oddoux [28]. (For example, Corollary 4.4.9 allows simple special cases of Propositions 4.4.3 and 4.4.4 that can easily be seen to apply also to nondeterministic automata; likewise, also the transition guard simplification rules given in Sect. 4.1 apply directly to nondeterministic automata.) Fritz [24] has investigated similar techniques based on the application of simulation relations to improve the nondeterminization construction of Miyano and Hayashi [49]. However, his techniques are not directly applicable to our construction, because they are based on simulation relations on alternating automata that are not allowed to have multiple acceptance conditions.

When working with linear alternating automata obtained from LTL formulas, where each state of the automaton corresponds to a known LTL formula, it may in some cases be possible to use the formula information for pruning states from a nondeterministic automaton during its construction. Due to the above correspondence and Corollary 5.3.4, each subautomaton of the nondeterministic automaton accepts the language corresponding to a conjunction of LTL formulas. Therefore, because every subautomaton that recognizes the empty language can always be removed from an alternating automaton without changing its language, it follows that we do not need to consider those states of the nondeterministic automaton that correspond to unsatisfiable conjunctions of LTL formulas. Borrowing techniques from direct translations of LTL formulas into nondeterministic automata [15, 31, 32], we can, for example, use simple syntactic checks to detect some of these unsatisfiable conjunctions to prune states from the nondeterministic automaton. More precisely, given a finite set of formulas $\Phi \subseteq LTL(AP)$ (corresponding to a set of states of the linear alternating automaton), the formula $\bigwedge_{\varphi\in\Phi}\varphi$ is easily seen to be unsatisfiable if one of the following conditions hold[1]:

- $\bot \in \Phi$;
- $\{\psi, \neg\psi\} \subseteq \Phi$ for some $\psi \in LTL(AP)$;
- $\big\{(\psi_1 \circ \psi_2), \neg\psi_1, \neg\psi_2\big\} \subseteq \Phi$ for some $\circ \in \{\vee, \mathsf{U_s}, \mathsf{U_w}\}$ and some $\psi_1, \psi_2 \in LTL(AP)$;
- $\big\{(\psi_1 \wedge \psi_2), \neg\psi_1\big\} \subseteq \Phi$ for some $\psi_1, \psi_2 \in LTL(AP)$; or
- $\big\{(\psi_1 \circ \psi_2), \neg\psi_2\big\} \subseteq \Phi$ for some $\circ \in \{\wedge, \mathsf{R_s}, \mathsf{R_w}\}$ and $\psi_1, \psi_2 \in LTL(AP)$.

We can try to improve heuristically the efficiency of the above tests by applying them to a set of formulas syntactically implied by the formulas in Φ instead of the set Φ itself; intuitively, explicating these syntactic implications increases the possibility of detecting the unsatisfiability of the conjunction using the above tests. Formally, we define the set $\mathrm{SI}(\Phi)$ of formulas syntactically implied by the elements of Φ as the smallest set of formulas that contains Φ as a subset and is closed under the following rules:

- If $\big\{(\psi_1 \circ \psi_2), \neg\psi_i\big\} \subseteq \mathrm{SI}(\Phi)$ for some $\circ \in \{\vee, \mathsf{U_s}, \mathsf{U_w}\}$, $\psi_1, \psi_2 \in LTL(AP)$ and $i \in \{1, 2\}$, then $\psi_{3-i} \in \mathrm{SI}(\Phi)$.
- If $(\psi_1 \wedge \psi_2) \in \mathrm{SI}(\Phi)$ for some $\psi_1, \psi_2 \in LTL(AP)$, then $\psi_1, \psi_2 \in \mathrm{SI}(\Phi)$.
- If $(\psi_1 \circ \psi_2) \in \mathrm{SI}(\Phi)$ for some $\circ \in \{\mathsf{R_s}, \mathsf{R_w}\}$, then $\psi_2 \in \mathrm{SI}(\Phi)$.

[1] We identify negated LTL formulas here with their positive normal forms.

It is straightforward to check using the semantics of LTL that for all finite sets of formulas $\Phi \subseteq LTL(AP)$, $\bigwedge_{\varphi\in\Phi} \varphi \equiv \bigwedge_{\varphi\in\mathrm{SI}(\Phi)} \varphi$. Therefore, if a state of the nondeterministic automaton corresponds to a set of formulas Φ, we can prune the state from the automaton if the formula $\bigwedge_{\varphi\in\mathrm{SI}(\Phi)} \varphi$ is unsatisfiable.

As a matter of fact, the sets of formulas syntactically implied by the conjunctions of formulas corresponding to the states of the nondeterministic automaton induce an equivalence relation between the states of the nondeterministic automaton, which suggests the possibility of using the relation for minimizing the nondeterministic automaton by quotienting (the minimization technique used with simulation relations in general [21, 22, 25]). However, it is a well-known fact that mere language equivalence between two subautomata of a nondeterministic automaton is not sufficient to guarantee that the automaton obtained by quotienting still accepts the same language (an example can be found, e.g., in the article by Somenzi and Bloem [62]). Therefore, the selection of representative states requires information also on the reachability between states in the nondeterministic automaton [62].

Finally, when implementing the language emptiness checks that arise during transition redundancy analysis (Sect. 4.4) via translation of alternating automata to nondeterministic automata, we are not usually interested in the actual nondeterministic automaton that accepts the intersection of the languages of one or more linear alternating automata, but only in the emptiness of the automaton. Furthermore, because all of our minimization techniques allow simplification only if the answer to the language emptiness question is positive, we always obtain a sound approximation of the emptiness checking result by assuming the automaton to be nonempty. This fact can be used to devise heuristics for dynamically allowing or disallowing some of the (potentially very costly) emptiness checks during minimization of alternating automata. For example, we could set a limit for the maximum size for any nondeterministic automaton to be used in an emptiness check and use this limit to abort the nondeterminization of alternating automata (with a negative answer to the emptiness question) if the automaton grows too large.

By caching the results of the emptiness checks for different combinations of alternating automata (encoded as a subset Q of states of an underlying alternating automaton $\mathcal{A}$ that contains all of the automata as its subautomata), we may be able to avoid the explicit construction of a nondeterministic automaton in later emptiness checks for other combinations of the automata. For example, it is easy to see that if $\bigcap_{q\in Q} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \emptyset$, then $\bigcap_{q\in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) = \emptyset$ for all $Q' \supseteq Q$; conversely, if $Q' \subseteq Q$, then $\bigcap_{q\in Q} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) \neq \emptyset$ implies that $\bigcap_{q\in Q'} \mathcal{L}_{\mathrm{fin}}(\mathcal{A}^q) \neq \emptyset$.

6 EMPTINESS CHECKING OF NONDETERMINISTIC AUTOMATA

Checking the emptiness of nondeterministic automata is easily reducible to a question on repeated reachability in automata [13, 76]. Therefore, the emptiness checking problem is solvable, for example, using time-efficient basic graph algorithms [65] to decide the existence of cycles satisfying certain acceptance constraints.

However, the practical difficulties to handle large automata, which are commonly encountered in model checking, have encouraged the study of algorithms that aim to improve the memory requirements of checking the emptiness of the automata. An alternative approach to emptiness checking was presented by Courcoubetis, Vardi, Wolper and Yannakakis [13], who introduced a linear-time on-the-fly emptiness checking algorithm (commonly known as the *nested depth-first search*) that is also compatible with common probabilistic explicit-state model checking techniques [35, 63, 79]. Further improvements and extensions to this algorithm were presented by several authors, including Godefroid and Holzmann [33], Holzmann, Peled, and Yannakakis [36], Edelkamp, Leue and Lluch-Lafuente [17], Brim, Černá and Nečesal [4], and Bošnački [2, 3].

Most known variants of the basic nested depth-first search algorithm, however, share the inability to handle automata with multiple acceptance conditions directly: to apply the nested depth-first search algorithm, the automata must first be "degeneralized" into an equivalent automaton with a classic Büchi acceptance condition [10, 13, 19, 28, 32]. Although a simple operation, the degeneralization may nevertheless increase the size of the automaton by a linear factor that depends on the number of acceptance conditions. Because this increase cannot be avoided in the general case [66], the transformation may thus decrease the optimal memory savings gained from the use of a special emptiness checking algorithm. Therefore, the potential advantages of avoiding degeneralization suggest applications for direct emptiness checking algorithms for automata with multiple acceptance conditions. Other advantages of such algorithms have previously been pointed out in the context of model checking under *fairness assumptions* (see, for example, Francez [23]) by Couvreur [14], who introduced an on-the-fly version of the algorithm of Tarjan [65] for the emptiness checking of automata with multiple acceptance conditions, and Bošnački [3], who proposed methods for combining the nested depth-first search with fairness and symmetry reduction.

In this section, we describe a variant [66] of the basic nested depth-first search emptiness checking algorithm. This variant of the algorithm can work directly on automata with multiple acceptance conditions and removes the need for degeneralization. Instead of using automata working in fin-acceptance mode, however, we now switch our focus to automata working in inf-acceptance mode to allow easy substitution of our algorithm for other constructions known from the literature. The intuition behind the nested depth-first search is also easier to explain using this acceptance mode. When dealing with nondeterministic automata obtained from linear alternating automata working in fin-acceptance mode, the simple correspondence between the acceptance modes (see Sect. 2.3.1) allows the mode change to be combined, for example, with the nondeterminization of linear alternating automata.

6.1 EMPTINESS CHECKING ALGORITHM

Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a nondeterministic automaton. Without loss of generality, we may assume that Δ contains no transitions with an empty guard (as noted in Sect. 4.1.2, these transitions do not affect the language of the automaton). The nonemptiness of the automaton is easily seen to be equivalent to the existence of a path from a state to itself that satisfies a constraint on the acceptance conditions associated with transitions connecting each state to its successor in the path [13, 76]. That is, $\mathcal{L}_{\text{inf}}(\mathcal{A}) \neq \emptyset$ holds iff $\mathcal{A}$ contains a path from its initial state q_I to a state that is reachable from itself via a nontrivial path in which the transitions connecting each state to its successor can be chosen such that the union of the transitions' acceptance conditions includes all elements of $\mathcal{F}$. We call these paths *accepting cycles* in the automaton.

It is easy to see that every cycle of $\mathcal{A}$ is accepting if $\mathcal{F} = \emptyset$, and thus the nonemptiness of $\mathcal{A}$ can in this case be decided simply by checking whether the initial state of the automaton is included in a path that visits some state of the automaton twice. To simplify the description of the general emptiness checking algorithm, we assume that $\mathcal{F}$ is nonempty for the rest of the discussion.

Outline of the Algorithm

The emptiness checking algorithm is based on the above correspondence between nonemptiness and the existence of an accepting cycle in the automaton. The algorithm (shown in pseudocode in Fig. 6.1) tries to find an accepting cycle in the automaton by traversing it using a depth-first search that drives another interleaved incremental search in the automaton. The main depth-first search uses a stack of states (the variable *path*), together with a set of states (*processed*) that stores the states that have already passed through the search stack. Intuitively, the *path* stack always contains a path from the initial state q_I to one of its descendants. In some occasions we treat the *path* stack as a set of states to check for the presence of elements in the stack; in practice, this information must be kept in an additional data structure (such as a hash table) to avoid a linear search through the stack each time such a check occurs.

The depth-first search processes the states of the automaton in depth-first search post-order. Before the algorithm backtracks from a state q currently on top of the depth-first search stack, the algorithm scans the transitions starting from q (line 11) for transitions with a nonempty set of acceptance conditions (we explain the role of the *conds* table below). For each such transition, the algorithm can start another search from the (unique) target state of the transition by calling the `markConditions` subroutine (lines 12–13) that marks the acceptance conditions of the transition to a set of states reachable from q. For this purpose, the algorithm uses a table *conds* that associates each state

Input: A nondeterministic automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ with $\mathcal{F} \neq \emptyset$ and $\Gamma \neq \emptyset$ for all $\langle q, \Gamma, F, Q' \rangle \in \Delta$.

Output: A report telling whether the automaton is inf-empty.

Initialize: $conds := [q_1 \mapsto \emptyset, \ldots, q_{|Q|} \mapsto \emptyset]$; $path := \emptyset$; $processed := \emptyset$;

```
checkEmptiness(⟨Σ, Q, Δ, q_I, 𝓕⟩: Nondeterministic automaton)
  begin
    path.push(q_I);
    while (path ≠ ∅) do begin
      q := path.top();
      while (∃q' ∈ Q \ (path ∪ processed) : q' is a successor of q) do
        begin
          path.push(q');
          q := q';
        end;
      for all ⟨q, Γ, F, {q'}⟩ ∈ Δ do
        if (F ∪ conds[q] ⊈ conds[q']) then
          markConditions(⟨Σ, Q, Δ, q_I, 𝓕⟩, {q'}, F ∪ conds[q]);
      if (conds[q] = 𝓕) then
        exit "Automaton is not inf-empty";
      processed := processed ∪ {q};
      path.pop();
    end;
    exit "Automaton is inf-empty";
  end;

markConditions(⟨Σ,Q, Δ, q_I, 𝓕⟩: Nondeterministic automaton; states ∈ 2^Q;
                conditions ∈ 2^𝓕)
  repeat
    remove any state q from states;
    conds[q] := conds[q] ∪ conditions;
    for all ⟨q, Γ, F, {q'}⟩ ∈ Δ do
      if (q' ∈ path ∪ processed and conditions ⊈ conds[q']) then
        states := states ∪ {q'};
  until (states = ∅);
```

Fig. 6.1: Algorithm for checking nondeterministic automata for inf-emptiness

of the automaton with a set of acceptance conditions. To avoid redundant work, the nested search is restricted to states $q' \in Q$ that have previously been encountered during the main depth-first search but for which $conds[q']$ has not yet been updated to include all acceptance conditions of some transition starting from q (lines 12 and 26).

Intuitively, if $f \in conds[q]$ holds for an acceptance condition $f \in \mathcal{F}$ and a state $q \in Q$, then the state q is the last state on some path, two states of which can be connected to each other with a transition including f in its acceptance conditions. This information is taken into account also whenever starting a nested search in the automaton (line 13): intuitively, if we already know that the state q belongs to a path, two states of which can be connected with a transition including f in its acceptance conditions, then all states reachable from q must also have this property.

Consider the contents of a set $conds[q]$ immediately before and after the loop between lines 11 and 13. If $f \notin conds[q]$ holds for some acceptance condition $f \in \mathcal{F}$ before the loop, but $f \in conds[q]$ holds after the loop, then it is easy to see from the operation of the markConditions subroutine that the automaton contains a cycle from the state q to itself such that it is possible to connect two states in this cycle with a transition including the acceptance condition f. Thus, for example, if $conds[q] = \emptyset$ holds before the loop, but $conds[q] = \mathcal{F}$ holds after the loop, then q is a common state of a collection of cycles "covering" each of the acceptance conditions. Thus the cycles can trivially be merged into an accepting cycle, and the automaton is nonempty, because q—being on top of the *path* stack—is obviously reachable from the initial state of the automaton. By the condition at line 14, the algorithm reports that the automaton is not inf-empty in this case as expected. The fact that the exit condition at line 14 is sound even if $conds[q]$ is nonempty before entering the loop at line 11 will be justified in the correctness proof presented in the end of this chapter.

Time and Space Complexity

Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ be a nondeterministic automaton ($\mathcal{F} \neq \emptyset$, $\Gamma \neq \emptyset$ for all $\langle q, \Gamma, F, Q \rangle \in \Delta$) given as input for the algorithm. Because the algorithm implements a depth-first search in the automaton, it is clear that the loop between lines 11 and 13 is executed once for each state (reachable from the initial state q_I) in the automaton. The loop between lines 6 and 10 can easily be implemented such that it is executed at most once for each transition in the automaton. If we assume that all set membership checks take constant time, the algorithm thus needs, using the standard "big-oh"-notation, $O\big(|Q| + c(|\mathcal{F}|) \cdot |\Delta|\big)$ steps plus the number of steps spent in the markConditions subroutine to execute, where $1 \leq c(|\mathcal{F}|) < \omega$ is a constant that depends (linearly) on the number of acceptance conditions in the automaton (required to account for the set inclusion test at line 12). By implementing set inclusion tests with basic bit vector operations, this constant is essentially equal to 1 if $|\mathcal{F}|$ does not exceed the word length of the underlying implementation architecture.

It is easy to see from the conditions on lines 12 and 26 of the algorithm that some set in the *conds* table is extended with a new element each time line 24 is executed. Therefore, the loop between lines 25 and 27 can executed at most $|\mathcal{F}|$ times for each state in the automaton, and thus all calls to the `markConditions` subroutine require $O\Big(|\mathcal{F}| \cdot \big(|Q| + c(|\mathcal{F}|) \cdot |\Delta|\big)\Big)$ steps. This is easily seen to also be the complexity of the running time of the entire algorithm. Thus, for any fixed number of acceptance conditions, the algorithm works in linear time in the total number of states and transitions in the automaton.

The *conds* table can be implemented using a simple hash table using $|Q| \cdot \big(s + |\mathcal{F}|\big)$ bits of storage, where s is the number of bits required for each key that is used for indexing states in the table. Information about the states currently on the *path* stack can be stored into the same hash table using an additional bit of memory per state. Although we could similarly add one more bit per state for storing the information about the states in the *processed* set, we can easily infer this information also from the presence of the state in the hash table and the hash table information, provided that the states are inserted into the table only as they are encountered during the main depth-first search. Thus, the algorithm needs, in addition to the memory required for the data structures needed for automaton traversal, a total of $|Q| \cdot \big(s + |\mathcal{F}| + 1\big)$ bits of memory. This memory limit is slightly lower than the theoretical worst-case memory consumption of the classic nested depth-first search algorithm when the degeneralization of the automata (mandatory for the classic algorithm) is taken into account [66].

Applications

The emptiness checking algorithm in Fig. 6.1 can be used both for checking the emptiness of automata that arise during language containment tests when simplifying alternating automata, and for the common task of model checking [13]. See the previous work of the author [66] for some further discussion and analysis, and a version of the algorithm that is, similar to the classic algorithm, compatible with common probabilistic model checking techniques such as *bitstate hashing* [35] and *hash compaction* [63, 79]. The generalized algorithm can also handle model checking under weak fairness assumptions. We note, however, that probabilistic methods cannot be used when checking the emptiness of automata for language containment tests, since the result that an automaton is only probably empty is not sufficient to guarantee the soundness of the language containment based simplification techniques discussed in Ch. 4.

6.2 CORRECTNESS OF THE ALGORITHM

In this section we show the correctness of the emptiness checking algorithm. Throughout the section, we refer to the components of a nondeterministic automaton $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ (with $\mathcal{F} \neq \emptyset$ and $\Gamma \neq \emptyset$ for all transitions $\langle q, \Gamma, F, Q' \rangle \in \Delta$) given as input for the algorithm. For all states $q \in Q$ in the automaton, we denote the (static) contents of the *path* stack and the *processed* set between lines 11 and 15 of the algorithm when q is on top of the *path* stack by $path(q)$ and $processed(q)$, respectively; clearly, the section of code between these lines is executed at most once for each state of the automaton.

Lemma 6.2.1 *Let $q \in Q$ be the state on top of the path stack between lines 11 and 15 of the algorithm. Then, q is a descendant of q' in $\mathcal{A}$ for all $q' \in path(q) \setminus \{q\}$.*

Proof: The algorithm uses *path* as the depth-first search stack; the states in the stack always form a path to the state currently on top of the stack. □

Lemma 6.2.2 *Let $q \in Q$ be a state on top of the path stack between lines 11 and 15 of the algorithm, and let $f \in \mathcal{F}$ be an acceptance condition. If $f \in conds[q]$ holds already before the algorithm enters the loop at line 11, then the automaton contains a nontrivial path from q to itself in which some state can be connected to its successor with a transition that includes f in its acceptance conditions.*

Proof: Because q is on top of the path stack, q cannot yet have been inserted into the set *processed*. Because $conds[q]$ was initially empty, the fact that $f \in conds[q]$ holds before the loop at line 11 implies that f was inserted into $conds[q]$ during a previous call to the `markConditions` subroutine with another state $q' \neq q$ on top of the *path* stack; it is clear from the calling convention and the operation of the subroutine that q is reachable from q' in the automaton. Furthermore, because the *processed* set grows monotonically during the execution of the algorithm, $q \notin processed(q)$ implies that $q \notin processed(q')$, and it follows that q must have been on the *path* stack when the `markConditions` subroutine was called in the state q', i.e., $q \in path(q')$. Therefore q' is also reachable from q in the automaton by Lemma 6.2.1, and the automaton contains a path from q to itself such that the path visits the state q'.

If q' is the start state of a transition including f in its acceptance conditions, then the result follows immediately if q was reached from q' during a nested search starting from the target state of one of these transitions. Otherwise (i.e., if no transition starting from q' includes f in its acceptance conditions, or if q was not reached from q' via any such transition), the fact that f was nevertheless added to $conds[q]$ while q' was on top of the *path* stack necessitates that f was already included in $conds[q']$ before the algorithm entered the loop at line 11 with q' on top of the *path* stack. Thus q' satisfies the assumption given in the lemma.

By repeating the above reasoning in q', we find a path from q' to itself through another state $q'' \neq q'$ (and $q'' \neq q$) such that f was added to $conds[q']$ while q'' was on top of the *path* stack. Using the same construction, we can now find a sequence of states $(q, q', q'', \ldots)$ in which every two successive states are reachable from each other in the automaton. Furthermore, because the states in this sequence are unique states of a finite automaton, the sequence is necessarily finite, and it ends with a pair of states $\hat{q}$ and q_f that can be connected to each other with transitions, one of which includes f in its acceptance conditions. Therefore, the path $(q, \ldots, q', \ldots, \hat{q}, \ldots, q_f, \ldots, \hat{q}, \ldots, q', \ldots, q)$ from q to itself has this same property. □

The following theorem establishes the soundness of the algorithm.

Theorem 6.2.3 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ ($\mathcal{F} \neq \emptyset$, $\Gamma \neq \emptyset$ for all $\langle q, \Gamma, F, Q' \rangle \in \Delta$) be a nondeterministic automaton given as input for the algorithm shown in Fig. 6.1. If the algorithm reports that $\mathcal{A}$ is not inf-empty, then $\mathcal{L}_{\mathrm{inf}}(\mathcal{A}) \neq \emptyset$.*

Proof: Assume that the algorithm reports that $\mathcal{A}$ is not inf-empty. Clearly, this can occur only if $\mathit{conds}[q] = \mathcal{F}$ holds at line 14 of the algorithm for some state $q \in Q$ currently on top of the *path* stack. Since $q_I \in \mathit{path}(q)$ is certainly true, there exists a (possibly trivial) path from q_I to q in the automaton by Lemma 6.2.1.

Let $f \in \mathcal{F}$ be an acceptance condition. If $f \in \mathit{conds}[q]$ holds before the algorithm enters the loop at line 11 (with q on top of the *path* stack), Lemma 6.2.2 proves the existence of a path from q to itself in which some transition can be chosen to include the acceptance condition f. Otherwise, if f was not included in $\mathit{conds}[q]$ before the loop, the algorithm had to add f to $\mathit{conds}[q]$ during a nested search starting from the state q itself. Therefore, q is the start state of a transition that includes f in its acceptance conditions and that can be taken along a path from q to itself. By repeating the consideration for all acceptance conditions, we can find a collection of paths from q to itself that can be merged into an accepting cycle of the automaton. By the discussion before the overview of the algorithm, this implies that $\mathcal{L}_{\mathrm{inf}}(\mathcal{A})$ is nonempty. $\square$

We now turn to the completeness of the algorithm. We begin by listing several additional basic properties of the algorithm.

Lemma 6.2.4 *Let $q \in Q$ be a state in the automaton. If the algorithm reaches line 11 with q on top of the path stack, then*

(a) *the algorithm will never start a nested search from any state $q' \in \mathit{processed}(q)$;*

(b) *if q' is a successor of q, then $q' \in \mathit{path}(q) \cup \mathit{processed}(q)$; and*

(c) *if there exists a state $q' \in \mathit{path}(q) \cup \mathit{processed}(q)$ that is reachable from q and has a successor $q'' \notin \mathit{path}(q) \cup \mathit{processed}(q)$, then $q' \in \mathit{path}(q)$.*

Proof:

(a) The main depth-first search visits each state of the automaton at most once. Since $q' \in \mathit{processed}(q)$, the search has already backtracked from q', and thus the algorithm cannot restart a nested search from q'.

(b) Immediate by the loop termination condition at line 6 of the algorithm.

(c) Let q' and q'' be states satisfying the requirements in the lemma. The case $q' = q$ is impossible by (b). We show that the assumption that $q' \in \mathit{processed}(q)$ leads to a contradiction, and therefore q' must be included in $\mathit{path}(q)$.

Assume that $q' \in \mathit{processed}(q)$, i.e., that the main search has already backtracked from q'. This implies that the algorithm has already passed through line 11 with q' on top of the *path* stack. But then, because q'' is a successor of q', it follows by (b) that $q'' \in path(q') \cup processed(q')$ held at this point. Because no states are ever removed from the *processed* set, $processed(q') \subseteq processed(q)$, and therefore $q'' \notin processed(q')$ necessarily holds by the choice of q''. Therefore $q'' \in path(q')$, which implies (again, by the choice of q'') that $q'' \in path(q') \setminus path(q)$. Therefore q'' must have been removed from the *path* stack before the algorithm reached line 11 with q on top of the *path* stack. However, this implies that $q'' \in processed(q)$, which is a contradiction. □

The completeness proof is based on an invariant of the nested search procedure. The invariant is stated using the following notion of *direct reachability*.

Let $q \in Q$ be a state on top of the *path* stack at lines 11–15. We say that the state $q' \in Q$ is *directly reachable* (from q) iff there exists a nontrivial path $(q_1, q_2, \ldots, q_n)$ $(2 \leq n < \omega)$ in the automaton such that $q_1 = q$, $q_n = q'$, $q_i \in processed(q)$ for all $1 \leq i \leq n$, and $\{q_2, q_3, \ldots, q_{n-1}\} \cap path(q) = \emptyset$. That is, q' is directly reachable from q if it is reachable from q via a nontrivial path contained in $processed(q)$ such that the path does not intersect $path(q)$ between its endpoints.

Lemma 6.2.5 *Let $q \in Q$ be a state in the automaton, and let $f \in \mathcal{F}$ be an acceptance condition included in $F \cup c[q]$ for some transition $t = \langle q, \Gamma, F, Q' \rangle \in \Delta$ starting from q at line 11 of the algorithm with q on top of the path stack, where $c[q]$ records the contents of $conds[q]$ immediately before the loop. Then, $f \in conds[q']$ holds for all states q' directly reachable from q via the transition t after the loop between lines 11 and 13.*

Proof: We proceed by induction on the length of paths starting from the state q via the transition t in the automaton. The case for t's unique target state (which is directly reachable from q by Lemma 6.2.4 (b)) is clear from the condition at line 12 and the operation of the markConditions subroutine.

Assume that the result holds for all states directly reachable from q via a shortest path with exactly n $(2 \leq n < \omega)$ states. Let q' be a state directly reachable from q via a shortest path $(q_1, q_2, \ldots, q_n, q')$ (where $q_1 = q$) with $n + 1$ states (we always assume that these paths begin with the transition t). Clearly, q_n is directly reachable from q via a path with n states. There are two cases:

1. The algorithm visits q_n during the call to the markConditions subroutine starting from t; clearly, $f \in conditions$ holds during this invocation of the subroutine. Because q' is a successor of q_n and directly reachable from q, the condition at line 26 of the algorithm guarantees that q' will be added to the *states* set if $f \notin conds[q']$. The termination condition of the subroutine (line 28) then ensures that the algorithm will not return from the subroutine until f has been added to $conds[q']$.

2. The algorithm does not visit q_n during the call to the markConditions subroutine starting from t. Because $f \in conds[q_n]$ nevertheless holds after the loop between lines 11 and 13 (by the induction hypothesis), f must have been added to $conds[q_n]$ during a previous call to

the subroutine. Assume that f was not included in $conds[q']$ when this search was started, and assume that this search did not visit the state q' (otherwise there is nothing to show). By the latter assumption, this search was started from another state $\hat{q} \neq q$ on top of the *path* stack. Because the search proceeded to q_n, $q_n \in path(\hat{q}) \cup processed(\hat{q})$; however, the fact that the search did not visit q' necessitates that $q' \notin path(\hat{q}) \cup processed(\hat{q})$, and thus q_n and $\hat{q}$ cannot be the same state (otherwise q' would be one of the successors of $\hat{q}$, all of which however belong to $path(\hat{q}) \cup processed(\hat{q})$ by Lemma 6.2.4 (b)). By Lemma 6.2.4 (c), it follows that $q_n \in path(\hat{q})$, and thus the algorithm can proceed to the loop at line 11 with q_n on top of the *path* stack only after backtracking from $\hat{q}$. On the other hand, this must nevertheless occur before the algorithm enters the loop with the state q on top of the *path* stack: because q' is directly reachable from q via the path $(q_1, q_2, \ldots, q_n, q')$ (where $q_1 = q$), it follows that $q_n \in processed(q)$, and thus, by Lemma 6.2.4 (a), the algorithm has already backtracked from q_n when it reaches line 11 with q on top of the *path* stack. Thus, because $f \in conds[q_n]$ holds when the algorithm enters the loop at line 11 with q_n on top of the *path* stack and q' is a successor of q_n, f will be added to the set $conds[q']$ during this loop if it is not there already. Thus $f \in conds[q']$ will hold also after the corresponding loop when q is on top of the *path* stack. This completes the induction. $\square$

We can now prove the completeness of the algorithm.

Theorem 6.2.6 *Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_I, \mathcal{F} \rangle$ ($\mathcal{F} \neq \emptyset$, $\Gamma \neq \emptyset$ for all $\langle q, \Gamma, F, Q' \rangle \in \Delta$) be a nondeterministic automaton given as input for the algorithm shown in Fig. 6.1. If $\mathcal{L}_{\text{inf}}(\mathcal{A}) \neq \emptyset$, then the algorithm reports that $\mathcal{A}$ is not inf-empty.*

Proof: Because $\mathcal{L}_{\text{inf}}(\mathcal{A}) \neq \emptyset$, $\mathcal{A}$ contains an accepting cycle that visits a descendant of the initial state q_I. Thus the automaton contains a *maximal nontrivial strongly connected component* (i.e., a maximal subset of states, all of which are reachable from each other in the automaton via a nontrivial path) reachable from the initial state q_I such that the component contains a cycle in which each state can be connected to its successor with transitions, the union of whose acceptance conditions includes all elements of $\mathcal{F}$.

Assume, contrary to the claim, that the algorithm reports that $\mathcal{A}$ is empty. In this case the main depth-first search will visit all states reachable from q_I. Let $C \subseteq Q$ be the first maximal strongly connected component of $\mathcal{A}$ entered in the main depth-first search among all components of $\mathcal{A}$ that contain an accepting cycle, and let $q \in C$ be the first state of C pushed on the *path* stack. Let $f \in \mathcal{F}$ be an acceptance condition. Because C contains an accepting cycle, there exists a state $q_f \in C$ that is connected to another state of the component with a transition $t \in \Delta$ that includes f in its acceptance conditions. By the choice of q, the main search will not backtrack from q until all states in C have been visited; therefore, the algorithm will enter the loop between lines 11 and 13 with q_f on top of the *path* stack while q is also still on the stack.

If q is at this point directly reachable from q_f via t, then $f \in \mathit{conds}[q]$ will hold after the loop by Lemma 6.2.5; this applies especially in the case $q_f = q$, since q is certainly directly reachable from itself via t (the target state of t is in C). Otherwise $q_f \neq q$, and q is not directly reachable from q_f via t when the loop is entered with q_f on top of the *path* stack. Let x be any simple nontrivial path from q_f to q via t. Clearly, any such path is contained in C. There are two cases:

- x is entirely contained in $\mathit{path}(q_f) \cup \mathit{processed}(q_f)$. Because q is not directly reachable from q_f via t, however, it follows from the definition of direct reachability that x visits a state in $\mathit{path}(q_f)$ strictly between its endpoints.

- x visits a state q'' that is not contained in $\mathit{path}(q_f) \cup \mathit{processed}(q_f)$. Because the path starts from the state q_f that is in $\mathit{path}(q_f)$, x contains a state $q' \in \mathit{path}(q_f) \cup \mathit{processed}(q_f)$ of which q'' is a successor. By Lemma 6.2.4 (c), it follows that $q' \in \mathit{path}(q_f)$. In addition, $q' \neq q_f$ holds by Lemma 6.2.4 (b), and $q' \neq q$, since q is the last state in the path and thus has no successor. Thus, x visits a state in $\mathit{path}(q_f)$ strictly between its endpoints also in this case.

The above analysis proves the existence of a state $\hat{q} \in \mathit{path}(q_f) \setminus \{q_f, q\}$ that is directly reachable from q_f via t. Because t includes the acceptance condition f, it follows by Lemma 6.2.5 that $f \in \mathit{conds}[\hat{q}]$ will hold after the loop between lines 11 and 13 when q_f is on top of the *path* stack. Because $\hat{q} \neq q_f$ and $\hat{q} \in \mathit{path}(q_f)$, the algorithm will reach this loop in the state $\hat{q}$ only after backtracking from q_f. On the other hand, by the choice of q, this will occur before the loop is started in the state q.

When the algorithm enters the loop at line 11 with $\hat{q}$ on top of the *path* stack, there are again two possibilities: either q is directly reachable from $\hat{q}$, in which case $f \in \mathit{conds}[q]$ will hold after the loop (by Lemma 6.2.5; $f \in \mathit{conds}(\hat{q})$ holds already before the loop), or there exists a state $\hat{q}' \in \big(\mathit{path}(\hat{q}) \cap C\big) \setminus \{\hat{q}, q\}$ that is directly reachable from $\hat{q}$, and $f \in \mathit{conds}[\hat{q}']$ holds after the loop.

By repeating the above reasoning if necessary, we are bound to find a state $\bar{q}$ from which q is directly reachable when the loop at line 11 is entered with $\bar{q}$ on top of the *path* stack. Therefore, f will be added to $\mathit{conds}[q]$ (at the latest) when the loop is executed in the state $\bar{q}$.

Since f is arbitrary, we can conclude that $\mathit{conds}[q] = \mathcal{F}$ will hold at line 14 when q is on top of the *path* stack. Hence, the algorithm reports that the automaton is not inf-empty, contrary to our assumption. This proves the completeness of the algorithm. $\square$

7 CONCLUSIONS, CRITICISMS AND OPEN QUESTIONS

The close connection between linear temporal logic and linear alternating automata gives rise to conceptually simple translation procedures between the logic and the automata. In practice, these automata are often expanded into nondeterministic automata to facilitate the implementation of decision procedures for the logic and its applications, using, for example, well-known simple graph algorithms. The worst-case exponential combinatorial cost of nondeterminization makes the efficiency of the decision procedures very sensitive to the properties of the automata; intuitively, even small changes to the automata can have significant effects on the size of the results of nondeterminization. This motivates the study of methods for optimizing the translation of LTL into linear alternating automata.

As seen from the results in Ch. 4, the translation benefits from a generalized definition for linear alternating automata that allows improving the translation, for example, with new translation rules for simplifying the transition structure of automata built from smaller components by exploiting structural properties of the components and language containment relationships between the components. Similar principles apply to removing transitions from the automata during the incremental construction. In many cases, the two-way connection between LTL and linear alternating automata allows the language containment checks to be implemented by reusing the basic translation procedure; in principle, the connection can also be used to devise a complementation procedure for linear alternating automata.

Heuristic simplification of the transitions of the automata increases the possibility of obtaining automata that can easily be modified into nondeterministic automata with only a constant increase in the size of the automata. Additionally, the translation reveals a syntactic subclass of LTL, the automata constructed from which provably have this property. In general, linear alternating automata can be translated into nondeterministic automata without exceeding the best known (2^n) worst-case upper bound for their number of states; however, the generalized definition of automata causes an increase in the number of generalized acceptance conditions of the nondeterministic automata. This increase can be countered somewhat by using a direct emptiness checking algorithm for generalized nondeterministic automata.

Nevertheless, the proposed constructions can also be criticized on some points:

- It can be argued that translating linear temporal logic into nondeterministic automata using alternating automata as an intermediate formalism introduces extra overhead to the translation procedure. Moreover, obtaining a practically competitive implementation of the procedure requires additional support from an optimized nondeterminization construction for alternating automata, if further explicit minimization of nondeterministic automata is to be avoided. However, due to some evidence that suggests the high difficulty of the task of implementing efficient translation procedures for LTL correctly in practice [67], the translation may be more manageable as two separate subtasks.

- The proposed procedure for translating LTL into alternating automata does not support on-demand construction of the automata, even though it would be very useful when combining the translation with on-the-fly nondeterminization or emptiness checking techniques [1, 31]. This feature is not unique to the proposed translation, however, as the need for constructing the full automaton is also implicit, for example, in the design of many simulation-based minimization techniques for alternating and nondeterministic automata. Nevertheless, on-the-fly techniques still remain applicable to nondeterminization of the automata and emptiness checking of the results. Moreover, because alternating automata are comparable in size to the LTL specifications, the full construction of the alternating automata does not as easily suffer from the exponential combinatorial cost that may occur when translating the specifications directly into nondeterministic automata.

- Many of the proposed structural optimizations to linear alternating automata depend on the explicit notion of a transition of an alternating automaton. Therefore, the optimizations are not applicable to the minimization of alternating automata in which the transition relation is given only implicitly as Boolean expressions. Handling the transitions explicitly also renders even the basic translation construction without any optimizations into an algorithm with both exponential space and time complexity (in the size of the LTL specification) in the worst case. Algorithms of comparable worst-case complexity are nevertheless common in all translation procedures that construct automata from LTL formulas explicitly; the practical effectiveness of all of these constructions relies on the assumption that the specifications given as input for the algorithms remain small.

- Despite the conceptual simplicity of enhancing the basic translation construction with language containment based optimizations, carefree use of the optimizations is bound to limit severely the practical effectiveness of any implementation without heuristics to restrict checking for optimization opportunities. Whether more sophisticated "global" approaches to minimization of alternating automata (for example, techniques based on using simulation relations [24, 25]) actually outperform such heavily restricted local optimizations in practice cannot obviously be evaluated without careful testing. The same holds for evaluating the relative efficiency of the various optimizations proposed for the construction. Additionally, the exact theoretical computational complexity of the language containment problem for the generalized automata requires further analysis.

The translation and automata minimization constructions presented in this work leave many open questions on their details. For example, the collection of refined translation rules presented in Sect. 4.3 is likely to be extensible with special cases also for the $\vee$ connective. Another question concerns a possible connection between the modified $\wedge$ rule and the nondeterminization of linear alternating automata: the reason behind the complex condition on acceptance conditions in the rule (see Example 4.3.1) is very similar to the reason why nondeterminization via a "subset construction" is impossible in general without introducing new acceptance conditions (see Example 5.3.2). This leaves the obvious question whether the translation rule for the $\wedge$ connective could be made more general at the cost of introducing new acceptance conditions in the application of the rule. More generally, a better understanding of the optimizations proposed to the translation (Ch. 4) requires careful analysis on their relative completeness, for example, whether the special cases for transition redundancy analysis (Sect. 4.4.2) are general enough for detecting all redundant initial self-loops of linear alternating automata.

Several authors [21, 60] have raised questions on whether the simple idea of translating logical specifications into as small nondeterministic automata as possible is actually valid for obtaining optimal reduction in combinatorial explosion in all practical applications. As a matter of fact, there is some research and experimental evidence [43, 60, 68] to support the view that the combinatorial explosion may be more effectively reduced in practice by replacing nondeterministic choice with deterministic choice in the automata, even though this change may increase the size of the automata. It is worth studying whether any of these ideas could be carried over to the construction of alternating automata from LTL formulas.

As noted already in Sect. 5.3, translating linear alternating automata into nondeterministic automata should not be done in practice without additional optimizations to nondeterminization. In comparison to previous constructions for translating linear alternating automata into plain nondeterministic automata (e.g., [28]), the construction presented in Sect. 5.3 has, for example, the awkward feature of breaking the intuitive one-to-one correspondence between the acceptance conditions of a linear alternating automaton and the strong temporal eventualities of an LTL formula from which the automaton was constructed. Although the increase in the number of acceptance conditions cannot be avoided in the general case, an obvious target for further study is whether the very restricted placement of the conditions in automata built from LTL formulas (see the discussion in Sect. 4.3.4) can be exploited to reduce the blow-up for these automata. On the other hand, it would be interesting to explore possible applications for the restricted syntactic subclass of LTL introduced in Sect. 5.1 and investigate the computational complexity of its decision and model checking problem, similar to the work of Demri and Schnoebelen [16].

On a more general note, a deeper examination of nondeterminization constructions for linear alternating automata could provide additional insight into the relationship between LTL and (in general, strictly more expressive) nondeterministic automata, with possible applications, for example, to the emptiness checking of these automata. Also, it should be investigated whether the nondeterminization construction for linear alternating automata could be extended into more general automata classes such as weak alternating automata.

ACKNOWLEDGMENTS

This report is a reprint of my Licentiate's thesis. I am deeply grateful to my instructor Keijo Heljanko for introducing me to the theoretical concepts and research problems behind my thesis, for his patience and for the effort he put on reading and commenting on the numerous drafts of this work. I would also like to thank my supervisor, Professor Ilkka Niemelä, for the opportunity to work on my thesis as a member of his research group in the Laboratory for Theoretical Computer Science at Helsinki University of Technology and for his invaluable advice and encouragement. I am grateful also to Tommi Junttila and Timo Latvala for several discussions and ideas. The financial support of Helsinki Graduate School in Computer Science and Engineering (HeCSE), Academy of Finland (Projects 47754 and 53695) and Nokia Foundation is gratefully acknowledged.

BIBLIOGRAPHY

[1] B. Bollig and M. Leucker. Deciding LTL over Mazurkiewicz traces. *Data & Knowledge Engineering*, 44(2):219–238, 2003.

[2] 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)*, number 2529 in Lecture Notes in Computer Science, pages 65–80. Springer-Verlag, 2002.

[3] D. Bošnački. A light-weight algorithm for model checking with symmetry reduction and weak fairness. In *Proceedings of the 10th* SPIN *Workshop on Model Checking of Software (SPIN 2003)*, volume 2648 of *Lecture Notes in Computer Science*, pages 89–103. Springer-Verlag, 2003.

[4] 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.

[5] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[6] J. A. Brzozowski and E. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[7] 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.

[8] 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.

[9] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[10] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.

[11] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceedings of the Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[12] E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. *Formal Methods in System Design*, 10(1):47–71, 1997.

[13] 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.

[14] 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.

[15] 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.

[16] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.

[17] S. Edelkamp, S. Leue, and A. Lluch-Lafuente. Direct explicit-state model checking in the validation of communication protocols. *International Journal on Software Tools for Technology Transfer (STTT)*, 2003. DOI: 10.1007/s10009-002-0104-3.

[18] 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.

[19] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984.

[20] 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.

[21] 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.

[22] K. Etessami, Th. Wilke, and R. A. Schuller. Fair simulation relations, parity games and state space reduction of 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.

[23] N. Francez. *Fairness*. Texts and Monographs in Computer Science. Springer-Verlag, 1986.

[24] 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.

[25] 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.

[26] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th Annual Symposium on Principles of Programming Languages (POPL 1980)*, pages 163–173. Association for Computing Machinery, 1980.

[27] P. Gastin, R. Meyer, and A. Petit. A (non-elementary) 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.

[28] 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.

[29] P. Gastin and D. Oddoux. LTL with past and two-way 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.

[30] M. C. W. Geilen. On the construction of monitors for temporal logic properties. *Electronic Notes in Theoretical Computer Science*, 55(2), 2001.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.

[36] G. J. Holzmann, D. Peled, and M. Yannakakis. On nested depth-first search. In *Proceedings of the 2nd* SPIN *Workshop*, volume 32 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.

[42] 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.

[43] T. Latvala. Efficient model checking of safety properties. In *Proceedings of the 10th* SPIN *Workshop on Model Checking of Software (SPIN 2003)*, volume 2648 of *Lecture Notes in Computer Science*, pages 74–88. Springer-Verlag, 2003.

[44] E. Leiss. Succinct representation of regular languages by Boolean automata. *Theoretical Computer Science*, 13:323–330, 1981.

[45] 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*, pages 97–107. Association for Computing Machinery, 1985.

[46] C. Löding. *Methods for the Transformation of Omega-Automata: Complexity and Connection to Second Order Logic*. Diploma thesis, Christian-Albrechts-University of Kiel, 1998.

[47] 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.

[48] 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.

[49] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[50] 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.

[51] D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata, the weak monadic theory of trees and its complexity. *Theoretical Computer Science*, 97:233–244, 1992.

[52] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[53] 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*, 141:69–107, 1995.

[54] 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.

[55] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[56] 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.

[57] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[58] G. S. Rohde. *Alternating Automata and the Temporal Logic of Ordinals*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.

[59] 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)*, number 2250 in Lecture Notes in Computer Science, pages 39–54. Springer-Verlag, 2001.

[60] R. Sebastiani and S. Tonetta. “More deterministic” vs. “smaller” Büchi automata for efficient LTL model checking. In *Proceedings of the 12th 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.

[61] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[62] 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.

[63] U. Stern and D. 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 1995)*, pages 333–348. Kluwer, 1996.

[64] R. E. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1/2):121–141, 1982.

[65] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[66] H. Tauriainen. Nested emptiness search for generalized Büchi automata. Research Report A79, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2003.

[67] 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.

[68] X. Thirioux. Simple and efficient translation from LTL formulas to Büchi automata. *Electronic Notes in Theoretical Computer Science*, 66(2), 2002.

[69] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, pages 133–191. Elsevier, 1990.

[70] W. Thomas. Languages, automata and logic. In *Handbook of Formal Languages*, volume III, pages 389–455. Springer-Verlag, 1997.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

[77] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

[78] 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*, number 2090 in Lecture Notes in Computer Science, pages 261–277. Springer-Verlag, 2001.

[79] 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.

[80] 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-A70	Petteri Kaski Isomorph-Free Exhaustive Generation of Combinatorial Designs. December 2001.
HUT-TCS-A71	Keijo Heljanko Combining Symbolic and Partial Order Methods for Model Checking 1-Safe Petri Nets. February 2002.
HUT-TCS-A72	Tommi Junttila Symmetry Reduction Algorithms for Data Symmetries. May 2002.
HUT-TCS-A73	Toni Jussila Bounded Model Checking for Verifying Concurrent Programs. August 2002.
HUT-TCS-A74	Sam Sandqvist Aspects of Modelling and Simulation of Genetic Algorithms: A Formal Approach. September 2002.
HUT-TCS-A75	Tommi Junttila New Canonical Representative Marking Algorithms for Place/Transition-Nets. October 2002.
HUT-TCS-A76	Timo Latvala On Model Checking Safety Properties. December 2002.
HUT-TCS-A77	Satu Virtanen Properties of Nonuniform Random Graph Models. May 2003.
HUT-TCS-A78	Petteri Kaski A Census of Steiner Triple Systems and Some Related Combinatorial Objects. June 2003.
HUT-TCS-A79	Heikki Tauriainen Nested Emptiness Search for Generalized Büchi Automata. July 2003.
HUT-TCS-A80	Tommi Junttila On the Symmetry Reduction Method for Petri Nets and Similar Formalisms. September 2003.
HUT-TCS-A81	Marko Mäkelä Efficient Computer-Aided Verification of Parallel and Distributed Software Systems. November 2003.
HUT-TCS-A82	Tomi Janhunen Translatability and Intranslatability Results for Certain Classes of Logic Programs. November 2003.
HUT-TCS-A83	Heikki Tauriainen On Translating Linear Temporal Logic into Alternating and Nondeterministic Automata. December 2003.

ISBN 951-22-6868-X
ISSN 1457-7615