Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 60
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 60
Espoo 2000 HUT-TCS-A60

A NEW UNFOLDING APPROACH TO LTL MODEL CHECKING

Javier Esparza and Keijo Heljanko

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 60
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 60
Espoo 2000 HUT-TCS-A60

A NEW UNFOLDING APPROACH TO LTL MODEL CHECKING

Javier Esparza and Keijo Heljanko

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 951-22-4999-5
ISSN 0783-5396

Picaset Oy
Helsinki 2000

ABSTRACT: A new unfolding approach to LTL model checking is presented, in which the model checking problem can be solved by direct inspection of a certain finite prefix. The techniques presented so far required to run an elaborate algorithm on the prefix.

KEYWORDS: Net unfoldings, model checking, tableau systems, LTL, Petri nets

Contents

1 INTRODUCTION

Unfoldings are a partial order technique for the verification of concurrent and distributed systems, initially introduced by McMillan [12, 13]. They can be understood as the extension to communicating automata of the well-known unfolding of a finite automaton into a (possibly infinite) tree. The unfolding technique can be applied to systems modelled by Petri nets, communicating automata, or process algebras [5, 4, 11]. It has been used to verify properties of circuits, telecommunication systems, distributed algorithms, and manufacturing systems [1].

Unfoldings have proved to be very suitable for deadlock detection and invariant checking [12, 13]. For these problems, one first constructs a so-called *complete prefix* [5], a finite initial part of the unfolding containing all the reachable states. This prefix is at most as large as the state space, and usually much smaller (often exponentially smaller). Once the prefix has been constructed, the deadlock detection problem can be easily reduced to a graph problem [12, 13], an integer linear programming problem [14], or to a logic programming problem [8].

In [3, 7] and [21, 20], unfolding-based model checking algorithms have been proposed for a simple branching-time logic and for LTL, respectively. Although the algorithms have been applied with success to a variety of examples, they are not completely satisfactory: After constructing the complete prefix, the model checking problem cannot be yet reduced to a simple problem like, say, finding cycles in a graph. In the case of LTL, which is the one considered in this paper, the intuitive reason is that the infinite sequences of the system are "hidden" in the finite prefix in a complicated way. In order to make them "visible", a certain graph has to be constructed. Unfortunately, the graph can be exponentially larger than the complete prefix itself.

Niebert has observed [15] that this exponential blow-up already appears in a system of n independent processes, each of them consisting of an endless loop with one single action as body. The complete prefix has size $\mathcal{O}(n)$, which in principle should lead to large savings in time and space with respect to an interleaving approach, but the graph is of size $\mathcal{O}(2^n)$, i.e. as large as the state space itself.

In this paper we present a different unfolding technique which overcomes this problem. Instead of unrolling the system until a complete prefix has been generated, we "keep on unrolling" for a while, and stop when certain conditions are met. There are two advantages:

- The model checking problem can be solved by a direct inspection of the prefix, and so we avoid the construction of the possibly exponential graph.
- The algorithm for the construction of the new prefix is similar to the old algorithm for the complete prefix; only the definition of a cut-off event needs to be changed.

The only disadvantage is the larger size of the new prefix. Fortunately, we are able to provide a bound: the prefix of a system with K reachable states contains at most $\mathcal{O}(K^2)$ events, assuming that the system is presented as a

1-safe Petri net or as a product of automata[1]. Notice that this is an upper bound: the new prefix is usually much smaller than the state space, and in particular for Niebert's example it grows linearly in n.

Another interesting point is that the new prefix can be seen as an extension to communicating automata of the tableau techniques for LTL (see for instance [22]). In order to emphasize the link to tableau systems we use tableau terminology, and speak of terminals, tableaux, soundness and completeness.

The paper is structured as follows. Section 2 introduces Petri net notations. Section 3 presents the automata theoretic approach to LTL model checking. In Sect. 4 the unfolding method is introduced. Sections 5 and 6 contain the tableau systems for the two subproblems. In Sect. 7 we show how LTL model checking can be solved with the presented tableau systems. In Sect. 8 we conclude and discuss topics for further research.

2 PETRI NETS

A *net* is a triple (P, T, F), where P and T are disjoint sets of *places* and *transitions*, respectively, and F is a function $(P \times T) \cup (T \times P) \to \{0, 1\}$. Places and transitions are generically called *nodes*. If $F(x, y) = 1$ then we say that there is an *arc* from x to y. The *preset* of a node x, denoted by ${}^\bullet x$, is the set $\{y \in P \cup T \mid F(y, x) = 1\}$. The *postset* of x, denoted by $x^\bullet$, is the set $\{y \in P \cup T \mid F(x, y) = 1\}$. In this paper we consider only nets in which every transition has a nonempty preset *and* a nonempty postset.

A *marking* of a net (P, T, F) is a mapping $P \to I\!N$ (where $I\!N$ denotes the natural numbers including 0). We identify a marking M with the multiset containing $M(p)$ copies of p for every $p \in P$. For instance, if $P = \{p_1, p_2\}$ and $M(p_1) = 1$, $M(p_2) = 2$, we write $M = \{p_1, p_2, p_2\}$.

A marking M *enables* a transition t if it marks each place $p \in {}^\bullet t$ with a token, i.e. if $M(p) > 0$ for each $p \in {}^\bullet t$. If t is enabled at M, then it can *fire* or *occur*, and its occurrence leads to a new marking M', obtained by removing a token from each place in the preset of t, and adding a token to each place in its postset; formally, $M'(p) = M(p) - F(p, t) + F(t, p)$ for every place p. For each transition t the relation $\xrightarrow{t}$ is defined as follows: $M \xrightarrow{t} M'$ if t is enabled at M and its occurrence leads to M'.

A 4-tuple $\Sigma = (P, T, F, M_0)$ is a *net system* if (P, T, F) is a net and M_0 is a marking of (P, T, F) (called the *initial marking* of Σ). A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$ is an *occurrence sequence* if there exist markings $M_1, M_2, \ldots, M_n$ such that

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots M_{n-1} \xrightarrow{t_n} M_n$$

M_n is the marking reached by the occurrence of σ, which is also denoted by $M_0 \xrightarrow{\sigma} M_n$. A marking M is a *reachable marking* if there exists an occurrence sequence σ such that $M_0 \xrightarrow{\sigma} M$. The *reachability graph* of a net system Σ is the labelled graph having the reachable markings of Σ as nodes, and the $\xrightarrow{t}$ relations (more precisely, their restriction to the set of

[1]More precisely, the number of non-cut-off events is at most $\mathcal{O}(K^2)$.

reachable markings) as edges. In this work we only consider net systems with finite reachability graphs.

A marking M of a net is *n-safe* if $M(p) \leq n$ for every place p. A net system Σ is n-safe if all its reachable markings are n-safe.

Labelled nets Let $\mathcal{L}$ be an alphabet. A *labelled net* is a pair (N, l) (also represented as a 4-tuple (P, T, F, l)), where N is a net and $l \colon P \cup T \to \mathcal{L}$ is a labelling function. Notice that different nodes of the net can carry the same label. We extend l to multisets of $P \cup T$ in the obvious way.

For each label $a \in \mathcal{L}$ we define the relation $\xrightarrow{a}$ between markings as follows: $M \xrightarrow{a} M'$ if $M \xrightarrow{t} M'$ for some transition t such that $l(t) = a$. For a finite sequence $w = a_1 a_2 \ldots a_n \in \mathcal{L}^*$, $M \xrightarrow{w} M'$ denotes that $M \xrightarrow{a_1} M_1 \xrightarrow{a_2} M_2 \ldots M_{n-1} \xrightarrow{a_n} M'$ holds for some reachable markings $M_1, M_2, \ldots, M_{n-1}$. For an infinite sequence $w = a_1 a_2 \ldots \in \mathcal{L}^\omega$, $M \xrightarrow{w}$ denotes that $M \xrightarrow{a_1} M_1 \xrightarrow{a_2} M_2 \ldots$ holds for some reachable markings $M_1, M_2 \ldots$.

The reachability graph of a labelled net system (N, l, M_0) is obtained by applying l to the reachability graph of (N, M_0). In other words, its nodes are the set

$$\{l(M) \mid M \text{ is a reachable marking}\}$$

and its edges are the set

$$\{l(M_1) \xrightarrow{l(t)} l(M_2) \mid M_1 \text{ is reachable and } M_1 \xrightarrow{t} M_2\} \,.$$

3 AUTOMATA THEORETIC APPROACH TO MODEL CHECKING LTL

We present how to modify the automata theoretic approach to model checking LTL [19] to best suit the net unfolding approach.

For technical convenience we use an action-based temporal logic instead of a state-based one, namely the linear temporal logic $tLTL'$ of Kaivola [10], which is immune to the stuttering of invisible actions (see [10, 9]). Given a finite set A of actions, and a set $V \subseteq A$ of visible actions, the abstract syntax of $tLTL'$ is given by:

$$\varphi ::= \top \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \mid \varphi_1 \,\mathcal{U}^a\, \varphi_2, \text{ where } a \in V$$

The semantics of $tLTL'$ is given as follows. Let φ be a $tLTL'$ formula, and $w = x_0\, x_1 \, \ldots \in A^\omega$. We denote by $w^{(i)}$ the suffix of w starting at x_i. We define that φ is *true* of w, denoted $w \models \varphi$, as follows:

- $w \models \top$ always.
- $w \models \neg\varphi$ iff not $w \models \varphi$.
- $w \models \varphi_1 \vee \varphi_2$ iff $w \models \varphi_1$ or $w \models \varphi_2$.
- $w \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff there is some $i \geq 0$, such that
  - $w^{(i)} \models \varphi_2$, and

- for all $0 \leq j < i$, $w^{(j)} \models \varphi_1$.

- $w \models \varphi_1 \, \mathcal{U}^a \, \varphi_2$ iff there is some $i \geq 0$, such that
  - $w^{(i)} = a \, \ldots$,
  - $w^{(i+1)} \models \varphi_2$,
  - for all $0 \leq j < i$, $w^{(j)} = a_j \, \ldots$ such that $a_j \neq a$, and
  - for all $0 \leq j < i$, $w^{(j)} \models \varphi_1$.

The semantics of $\varphi_1 \, \mathcal{U} \, \varphi_2$ is as expected. Loosely speaking, a sequence w satisfies $\varphi_1 \, \mathcal{U}^a \, \varphi_2$ if φ_1 holds until the first a in w, and then φ_2 holds.

Given a net system $\Sigma = (P, T, F, l, M_0)$, where all the transitions of Σ are labelled with actions from the set A (and thus $A \subseteq \mathcal{L}$), and a formula φ of $tLTL'$, the model checking problem consists of deciding if all the infinite firing sequences[2] of Σ satisfy φ. Namely, given a labelled net system $\Sigma = (P, T, F, l, M_0)$ we define that $\Sigma \models \varphi$ if and only if $w \models \varphi$ for all $w \in A^\omega$ such that $M_0 \xrightarrow{w}$.

The *automata theoretic approach* attacks this problem as follows. First, a procedure similar to that of [6] converts the *negation* of φ into a Büchi automaton (definition to follow) $\mathcal{A}_{\neg\varphi}$ over the alphabet $\Gamma = V \cup \{\tau\}$, where $\tau \notin A$ is a new label used to represent all the invisible actions. More precisely, we introduce a function $v : A \rightarrow V \cup \{\tau\}$ which is defined as follows: for all $a \in V : v(a) = a$, and for all $a \in A \setminus V : v(a) = \tau$. We will use the fact that an infinite sequence $w \in A^\omega$ satisfies the $tLTL'$ formula φ if and only if the Büchi automaton $\mathcal{A}_\varphi$ accepts the sequence w', where w' is the image of w under v.

A *Büchi automaton* $\mathcal{A}$ is a tuple $(\Gamma, Q, q^0, \rho, F)$, where Γ is an *alphabet*, Q is a finite nonempty set of *states*, $q^0 \in Q$ is an *initial state*, $F \subseteq Q$ is the set of *accepting states*, and $\rho \subseteq Q \times \Gamma \times Q$ is the *transition relation*. As usual, $\mathcal{A}$ accepts an infinite word $w \in \Gamma^\omega$ if some run of $\mathcal{A}$ on w visits some state in F infinitely often.

Given a labelled net system Σ, and the Büchi-automaton $\mathcal{A}_{\neg\varphi}$ we can define a product Büchi-automaton $\mathcal{P}$, whose runs correspond to infinite runs of the net system Σ which do not satisfy the formula φ. (See Appendix A.1.1.) The language of the automaton $\mathcal{P}$ will thus be non-empty iff $\Sigma \not\models \varphi$. We denote by $|RG|$ the number of reachable markings of Σ, and by $|Q|$ the number of states of $\mathcal{A}_{\neg\varphi}$. The number of states of $\mathcal{P}$ is bounded by $(|RG| \cdot |Q|) + 1$, and will thus be of size $\mathcal{O}(|RG|)$, when the formula φ is fixed.

The problem with using this approach for our purposes is that to implement it, the automaton $\mathcal{A}_{\neg\varphi}$ needs to be synchronized with the net system Σ on all transitions (visible or not). This effectively sequentializes all the behaviors of the product system and thus disables all the benefits of the net unfolding method when applied to the product system. (Net unfoldings require concurrency in order to obtain space savings when compared to the interleaving approach.)

Therefore we will propose a different approach, in which the net system will only synchronize with the visible actions of the property automaton, and

[2]This is a small technical difference to Kaivola's semantics which considers both finite and infinite behaviors of the system.

handle the cases in which the system does invisible moves separately. Our approach is similar, however, not identical in all technical details, to Valmari's tester approach [18].

Synchronization. We define the synchronization of a labelled net system and a Büchi automaton[3]. Let $\Sigma = (P, T, F, l, M_0)$, where the transitions of T are labelled with letters taken from a set A of actions containing a subset V of visible actions. Let $\mathcal{A}_{\neg\varphi} = (V \cup \{\tau\}, Q, q^0, \rho, F')$, i.e., the alphabet of $\mathcal{A}_{\neg\varphi}$ contains the visible actions plus the special action τ. For technical reasons, we first add a new initial state $\hat{q}$ to Q, a new visible action $\hat{a}$ to V, and a new transition $\hat{q} \xrightarrow{\hat{a}} q^0$ to ρ. Similarly, we add a new place $\hat{p}$ to P, and a new transition $\hat{t}$ to T, labelled by $\hat{a}$. This new transition has $\hat{p}$ as preset, and it generates the initial marking M_0 when it fires. The new initial marking is $\hat{M} = \{\hat{p}\}$. Basically, after the modification Σ first fires a visible transition labelled with $\hat{a}$ and then behaves as before.

We keep Σ and $\mathcal{A}_{\neg\varphi}$ as names for the results of the modifications. Now, the *synchronization* $\Sigma_{\neg\varphi}$ is constructed in a sequence of steps:

- start with Σ;
- add a new place for each state of Q, and mark the place for $\hat{q}$ with one token;
- for each $t \in T$ labelled by a *visible* action $a \in V$, do the following:
  - for each transition $u\colon q \xrightarrow{a} q'$ of ρ, add a new transition (t, u) to $\Sigma_{\neg\varphi}$, labelled by a, with preset ${}^\bullet t \cup \{q\}$ and postset $t^\bullet \cup \{q'\}$;
  - after all these transitions have been added, remove t together with all its adjacent arcs.

Observe that the transitions of Σ labelled by invisible actions are not touched by the synchronization. Usually, we expect this set of transitions to be large, in which case the synchronization does not destroy much of the concurrency of Σ.

We define two sets of transitions of $\Sigma_{\neg\varphi}$:

- The set I of *infinite trace monitors* contains the transitions (t, u) such that $u\colon q \xrightarrow{a} q'$ for some final state q' of $\mathcal{A}_{\neg\varphi}$. Loosely speaking, these are the transitions which put a token in a final state of the Büchi automaton
- The set L of *livelock monitors* contains the transitions (t, u) such that $u\colon q \xrightarrow{a} q'$ for some state q' satisfying the following condition: with q' as initial state, the automaton $\mathcal{A}_{\neg\varphi}$ accepts an infinite sequence of invisible transitions (a *livelock*).

We have the following result (see Appendix A.1):

Proposition 1 *The synchronization $\Sigma_{\neg\varphi}$ has the following properties: None of its reachable markings enable two I-transitions concurrently, and none of its reachable markings enable a L-transition and a V-transition concurrently.*

[3] We give a somewhat informal but hopefully precise definition. A formal definition is longer and more obscure.

3.1 From emptiness checking to illegal ω-traces and illegal livelocks

We now reduce the model checking problem for a formula φ to two simpler problems on the synchronization $\Sigma_{\neg\varphi}$. Let Σ be a net system, where T is divided into two sets V and $T \setminus V$ of *visible* and *invisible* transitions. Moreover, T contains two special subsets L and I. An *illegal ω-trace* of Σ is an infinite sequence $M_0 \xrightarrow{t_1 t_2 \dots}$ such that $t_i \in I$ for infinitely many indices i. An *illegal livelock* of Σ is an infinite sequence $M_0 \xrightarrow{t_1 t_2 \dots t_i} M \xrightarrow{t_{i+1} t_{i+2} \dots}$ such that $t_i \in L$, and $t_{i+k} \in (T \setminus V)$ for all $k \geq 1$.

We have the following result (see Appendix A.1.1):

Theorem 2 *Let Σ be a labelled net system, and φ a tLTL$'$-formula. $\Sigma \models \varphi$ if and only if $\Sigma_{\neg\varphi}$ has no illegal ω-traces and no illegal livelocks.*

The intuition behind this theorem is as follows. Assume that Σ can execute an infinite firing sequence corresponding to a word $w \in (V \cup \{\tau\})^\omega$ violating φ (where 'corresponding' means that the firing sequence executes the same visible actions in the same order, and an invisible action for each τ). If w contains infinitely many occurrences of visible actions, then $\Sigma_{\neg\varphi}$ contains an illegal ω-trace; if not, it contains an illegal livelock.

3.2 Setting and running example

In the next sections we provide unfolding-based solutions to the problems of detecting illegal ω-traces and illegal livelocks. We solve the problems in an abstract setting. We fix a net system $\Sigma = (P, T, F, M_0)$, where T is divided into two sets V and $T \setminus V$ of *visible* and *invisible* transitions, respectively. Moreover, T contains two special subsets L and I. We assume that no reachable marking of Σ concurrently enables a transition of V and a transition of L. We further assume that M_0 is 1-safe. In particular, when applying the results to the model checking problem for *tLTL$'$* and Petri nets, the system Σ is the synchronization $\Sigma_{\neg\varphi}$ of a Petri net and a Büchi automaton, and it satisfies these conditions.

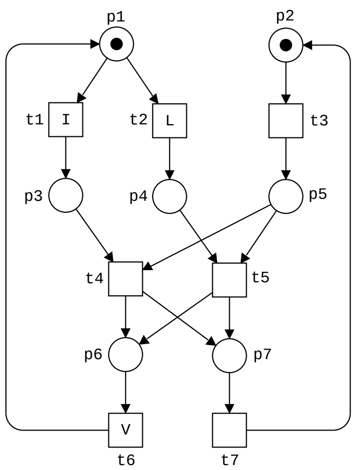

Figure 1: A net system

We use as running example the net system of Fig. 1. We have $V = \{t_6\}$, $I = \{t_1\}$, and $L = \{t_2\}$. The system has illegal ω-traces (for instance, $(t_1 t_3 t_4 t_6 t_7)^\omega$), but no illegal livelocks.

4 BASIC DEFINITIONS ON UNFOLDINGS

In this section we briefly introduce the definitions we need to describe the unfolding approach to our two problems. More details can be found in [5].

Occurrence nets. Given two nodes x and y of a net, we say that x is *causally related* to y, denoted by $x \leq y$, if there is a (possibly empty) path of arrows from x to y. We say that x and y are in *conflict*, denoted by $x \# y$, if there is a place z, different from x and y, from which one can reach x and y, exiting z by different arrows. Finally, we say that x and y are *concurrent*, denoted by $x \; co \; y$, if neither $x \leq y$ nor $y \leq x$ nor $x \# y$ hold. A *co-set* is a set of nodes X such that $x \; co \; y$ for every $x, y \in X$. *Occurrence nets* are those satisfying the following three properties: the net, seen as a graph, has no cycles; every place has at most one input transition; and, no node is in self-conflict, i.e., $x \# x$ holds for no x. A place of an occurrence net is *minimal* if it has no input transitions. The net of Fig. 2 is an infinite occurrence net with minimal places a, b. The *default initial marking* of an occurrence net puts one token on each minimal place an none in the rest.

Branching processes. We associate to Σ a set of *labelled* occurrence nets, called the *branching processes* of Σ. To avoid confusions, we call the places and transitions of branching processes *conditions* and *events*, respectively. The conditions and events of branching processes are labelled with places and transitions of Σ, respectively. The conditions and events of the branching processes are subsets from two sets $\mathcal{B}$ and $\mathcal{E}$, inductively defined as the smallest sets satisfying the following conditions:

- $\bot \in \mathcal{E}$, where $\bot$ is an special symbol;
- if $e \in \mathcal{E}$, then $(p, e) \in \mathcal{B}$ for every $p \in P$;
- if $\emptyset \subset X \subseteq \mathcal{B}$, then $(t, X) \in \mathcal{E}$ for every $t \in T$.

In our definitions of branching process (see below) we make consistent use of these names: The label of a condition (p, e) is p, and its unique input event is e. Conditions $(p, \bot)$ have no input event, i.e., the special symbol $\bot$ is used for the minimal places of the occurrence net. Similarly, the label of an event (t, X) is t, and its set of input conditions is X. The advantage of this scheme is that a branching process is completely determined by its sets of conditions and events. We make use of this and represent a branching process as a pair (B, E).

Definition 3 *The set of* finite branching processes *of a net system Σ with the initial marking $M_0 = \{p_1, \ldots, p_n\}$ is inductively defined as follows:*

- $(\{(p_1, \bot), \ldots, (p_n, \bot)\}, \emptyset)$ *is a branching process of* Σ.[4]
- *If* (B, E) *is a branching process of* Σ, $t \in T$, *and* $X \subseteq B$ *is a co-set labelled by* ${}^\bullet t$, *then* $(B \cup \{(p, e) \mid p \in t^\bullet\} \ , \ E \cup \{e\})$ *is also a branching process of* Σ, *where* $e = (t, X)$. *If* $e \notin E$, *then* e *is called a* possible extension *of* (B, E).

The set of branching processes of Σ is obtained by declaring that the union of any finite or infinite set of branching processes is also a branching process, where union of branching processes is defined componentwise on conditions and events. Since branching processes are closed under union, there is a unique maximal branching process, called the *unfolding* of Σ. The unfolding of our running example is an infinite occurrence net. Figure 2 shows an initial part. Events and conditions have been assigned identificators that will be used in the examples. For instance, the event $(t_1, \{(p_1, \bot)\})$ is assigned the identificator 1.

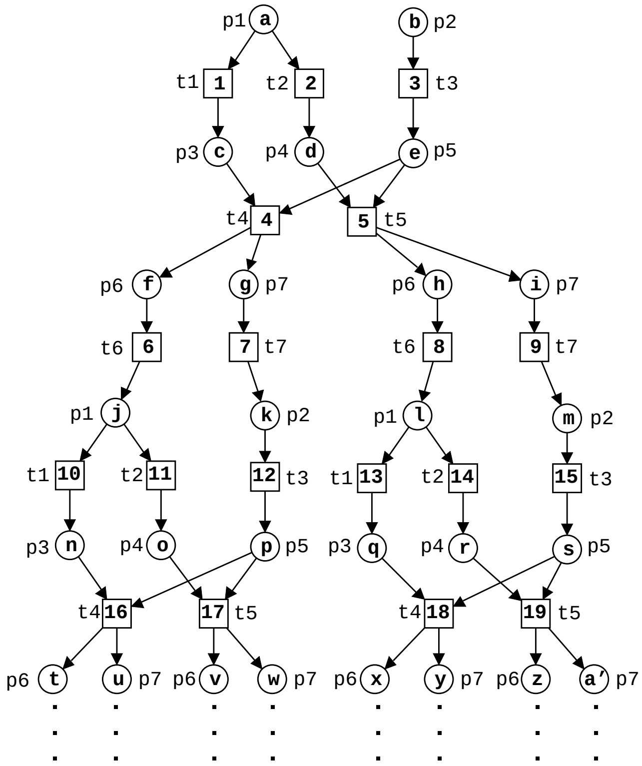

Figure 2: The unfolding of Σ

We take as partial order semantics of Σ its unfolding. This is justified, because it can be easily shown the reachability graphs of Σ and of its unfolding coincide. (Notice that the unfolding of Σ is a *labelled* net system, and so its reachability graph is defined as the *image* under the labelling function of the reachability graph of the *unlabelled* system.)

Configurations. A *configuration* of an occurrence net is a set of events C satisfying the two following properties: C is causally closed, i.e., if $e \in C$ and $e' < e$ then $e' \in C$, and C is conflict-free, i.e., no two events of C

[4] This is the point at which we use the fact that the initial marking is 1-safe.

are in conflict. Given an event e, we call $[e] = \{e' \in E \mid e' \leq e\}$ the *local configuration* of e. Let Min denote the set of minimal places of the branching process. A configuration C of the branching process is associated with a marking of Σ denoted by $Mark(C) = l((Min \cup C^\bullet) \setminus {}^\bullet C)$.

In Fig. 2, $\{1,3,4,6\}$ is a configuration, and $\{1,4\}$ (not causally closed) or $\{1,2\}$ (not conflict-free) are not. A set of events is a configuration if and only if there is one or more firing sequences of the occurrence net (from the default initial marking) containing each event from the set exactly once, and no further events. These firing sequences are called *linearisations*. The configuration $\{1,3,4,6\}$ has two linearisations, namely $1\,3\,4\,6$ and $3\,1\,4\,6$. All linearisations lead to the same reachable marking. For example, the two sequences above lead to the marking $\{p_1, p_7\}$. By applying the labelling function to a linearisation we obtain a firing sequence of Σ. Abusing of language, we also call this firing sequence a linearisation. In our example we obtain $t_1t_3t_4t_6$ and $t_3t_1t_4t_6$ as linearisations.

Given a configuration C, we denote by $\uparrow C$ the set of events $e \in E$, such that: (1) $e' < e$ for some event $e' \in C$, and (2) e is not in conflict with any event of C. Intuitively, $\uparrow C$ corresponds to the behavior of Σ from the marking reached after executing any of the linearisations of C. We call $\uparrow C$ the *continuation* after C of the unfolding of Σ. If C_1 and C_2 are two finite configurations leading to the same marking, i.e. $Mark(C_1) = M = Mark(C_2)$, then $\uparrow C_1$ and $\uparrow C_2$ are isomorphic, i.e., there is a bijection between them which preserves the labelling of events and the causal, conflict, and concurrency relations (see [5]).

5 A TABLEAU SYSTEM FOR THE ILLEGAL ω-TRACE PROBLEM

In this section we present an unfolding technique for detecting illegal ω-traces. We introduce it using the terminology of tableau systems, the reason being that the technique has many similarities with tableau systems as used for instance in [22] for model-checking LTL, or in [16] for model-checking the mu-calculus. In order to exhibit the similarities, we think of the conditions of a branching process as facts, and of its events as *inference rules*: if the facts represented by the input conditions of an event hold simultaneously (i.e., form a *co*-set), then the inference rule can be applied and the facts represented by the output conditions can be inferred. In this way, a branching process can be seen as a proof tree. As in [16] or [2], we introduce a notion of terminal inference, i.e., of an inference which marks the end of a particular branch of the proof tree; we also define successful and unsuccessful terminals.

In order to define the terminals of the tableau systems we need the notion of *adequate order* on configurations, first introduced in [5]. In fact, our tableau system will be parametric in the adequate order, i.e., we will obtain a different system for each adequate order.

5.1 Adequate orders

Given a configuration C of the unfolding of Σ, we denote by $C \oplus E$ the set $C \cup E$, under the condition that $C \cup E$ is a configuration satisfying $C \cap E = \emptyset$. We say that $C \oplus E$ is an *extension* of C. Now, let C_1 and C_2 be two finite configurations leading to the same marking. Then $\uparrow C_1$ and $\uparrow C_2$ are isomorphic. This isomorphism, say f, induces a mapping from the extensions of C_1 onto the extensions of C_2; the image of $C_1 \oplus E$ under this mapping is $C_2 \oplus f(E)$.

Definition 4 *A partial order $\prec$ on the finite configurations of the unfolding of a net system is an* adequate order *if:*

- *$\prec$ is well-founded,*
- *$C_1 \subset C_2$ implies $C_1 \prec C_2$, and*
- *$\prec$ is preserved by finite extensions; if $C_1 \prec C_2$ and $Mark(C_1) = Mark(C_2)$, then the isomorphism f from above satisfies $C_1 \oplus E \prec C_2 \oplus f(E)$ for all finite extensions $C_1 \oplus E$ of C_1.*

Total adequate orders are particularly good for our tableau systems because they lead to stronger conditions for an event to be a terminal, and so to smaller tableaux. Total adequate orders for 1-safe Petri nets and for synchronous products of transition systems, have been presented in [5, 4].

5.2 The tableau system

Given a configuration C of the unfolding of Σ, denote by $\#_I C$ the number of events $e \in C$ labelled by transitions of I.

Definition 5 *An event e of a branching process BP is a* repeat *(with respect to $\prec$) if BP contains another event e', called the* companion *of e, such that $Mark([e']) = Mark([e])$, and either*

(I) $e' < e$, or

(II) $\neg(e' < e)$, $[e'] \prec [e]$, *and* $\#_I[e'] \geq \#_I[e]$.

A terminal *is a minimal repeat with respect to the causal relation; in other words, a repeat e is a terminal if the unfolding of Σ contains no repeat $e' < e$. Repeats, and in particular terminals, are of type I or type II, according to the condition they satisfy.*

Events labelled by I-transitions are called I-events. A repeat e with companion e' is successful *if it is of type I, and $[e] \setminus [e']$ contains some I-event. Otherwise it is* unsuccessful.

A tableau *is a branching process BP such that for every possible extension e of BP at least one of the immediate causal predecessors of e is a terminal. A tableau is* successful *if at least one of its terminals is successful.*

Loosely speaking, a tableau is a branching process which cannot be extended without adding a causal successor to a terminal. In the case of a terminal of type I, $\uparrow[e]$ need not be constructed because $\uparrow[e']$, which is isomorphic to it, will be in the tableau. In the case of a terminal of type II, $\uparrow[e]$ need not be constructed either, because $\uparrow[e']$, which is isomorphic to it, will appear in the tableau. However, in order to guarantee completeness, we need the condition $\#_I[e'] \geq \#_I[e]$.

The tableau construction is simple. Given $\Sigma = (N, M_0)$, where $M_0 = \{p_1, \ldots, p_n\}$, start from the branching process $(\{(p_1, \perp), \ldots, (p_n, \perp)\}, \emptyset)$. Add events according to the inductive definition of branching process, but with the restriction that no event having a terminal as a causal predecessor is added. Events are added in $\prec$ order; more precisely, if $[e] \prec [e']$, then e is added before e'. The construction terminates when no further events can be added.

Figure 3 shows the tableau corresponding to the net system of Fig. 1, using the total adequate order of [5]. (We can also take the order of [4], which for the examples yields the same results.) All we need to know about this order is that for the events 4 and 5 in Fig. 2, $[4] \prec [5]$ holds.

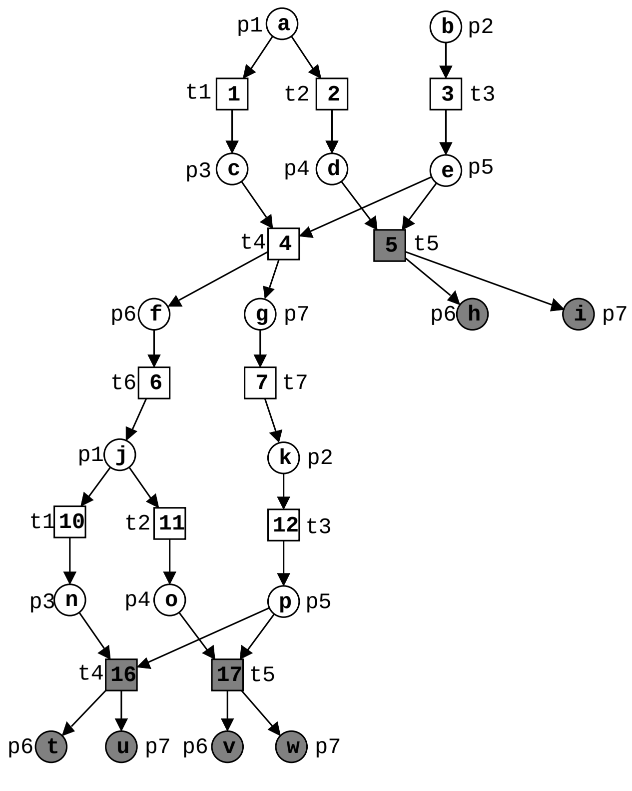

Figure 3: The tableau system for the illegal ω-trace problem.

Events 16 and 17 are terminals of type I having event 4 as companion. Event 16 is successful because the set $[16] \setminus [4] = \{6, 7, 10, 11, 12, 16\}$ contains an I-event, namely 10. The intuition behind these terminals is rather clear: a terminal of type I corresponds to a cycle in the reachability graph. Loosely speaking, the events of $[16] \setminus [4]$ correspond to a firing sequence leading from $Mark([4])$ to $Mark([16])$, and these two markings coincide. Since $[16] \setminus [4]$ contains an I-event, the firing sequence contains a transition of I, and so we have found an illegal ω-trace. The set $[17] \setminus [4]$ doesn't contains

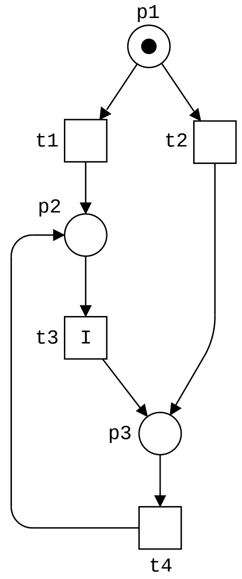

Figure 4: A net system

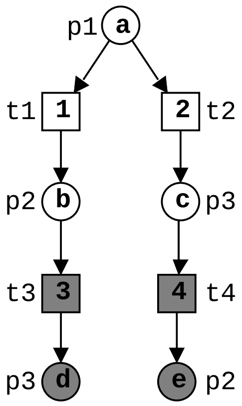

Figure 5: Tableau without the condition $\#_I[e'] \geq \#_I[e]$

any I-event, but $\uparrow[17]$ need not be constructed, because it is isomorphic to $\uparrow[4]$. Event 5 is a terminal of type II with event 4 as companion because $Mark([4]) = \{p_6, p_7\} = Mark([5])$, $[4] \prec [5]$, and $1 = \#_I[4] \geq \#_I[5] = 0$. The intuition is that $\uparrow[5]$ need not be constructed, because it is isomorphic to $\uparrow[4]$.

However, this doesn't explain why the condition $\#_I[e'] \geq \#_I[e]$ is needed. The following example shows that after removing this condition the tableau system is no longer complete. The net system of Fig. 4 has an illegal ω-trace. The tableau without the condition $\#_I[e'] \geq \#_I[e]$ is shown in Fig. 5. Event 3 is a terminal with event 2 as companion (the condition $\#_I[e'] \geq \#_I[e]$ has been removed!); event 4 is a terminal with event 1 as companion. Both terminals are unsuccessful.

Let K denote the number of reachable markings of Σ, and let B denote the maximum number of tokens that the reachable markings of Σ put in all the places of Σ.

We have the following result (see Appendix A.2):

Theorem 6 *Let $\mathcal{T}$ be a tableau of Σ constructed according to a total adequate order $\prec$.*

- $\mathcal{T}$ *is successful if and only if* Σ *has an illegal* ω*-trace.*
- $\mathcal{T}$ *contains at most* $K^2 \cdot B$ *non-terminal events.*
- *If the transitions of* I *are pairwise non-concurrent, then* $\mathcal{T}$ *contains at most* K^2 *non-terminal events.*

6 A TABLEAU SYSTEM FOR THE ILLEGAL LIVELOCK PROBLEM

The tableau system for the illegal livelock problem is a bit more involved that that of the illegal ω-trace problem. In a first step we compute a set $CP = \{M_1, \ldots, M_n\}$ of reachable markings of Σ, called the set of *checkpoints*. This set has the following property: if Σ has an illegal livelock, then it also has an illegal livelock $M_0 \xrightarrow{t_1 t_2 \ldots t_i} M \xrightarrow{t_{i+1} t_{i+2} \cdots}$ such that $t_i \in L$ and M is a checkpoint. For the computation of CP we use the unfolding technique of [5] or [4]; the procedure is described in Sect. 6.1.

The tableau system solves the problem whether some checkpoint enables an infinite sequence of invisible actions. Clearly, Σ has an illegal livelock if and only if this is indeed the case. For this, we consider the net N_{inv} obtained from N by removing all the visible transitions together with their adjacent arcs. We construct unfoldings for the net systems $(N_{inv}, M_1), \ldots, (N_{inv}, M_n)$, and check on them if the systems exhibit some infinite behavior. The tableau system is described in Sect. 6.2.

6.1 Computing the set of checkpoints.

We construct the complete prefix of the unfolding of Σ as defined in [5] or [4]. In the terminology of this paper, the complete prefix corresponds to a tableau in which an event e is a terminal if there is an event e' such that $Mark([e']) = Mark([e])$, and $[e'] \prec [e]$. As shown in [5, 4], the reachability graph of Σ and of its complete prefix coincide.

Definition 7 *A marking* M *belongs to the set* CP *of* checkpoints *of* Σ *if* $M = Mark([e])$ *for some non-terminal event* e *of the complete prefix of* Σ *labelled by a transition of* L.

Let us compute CP for our example. The complete prefix of Σ coincides with the tableau for the illegal ω-trace problem in Fig. 3. The events labelled by t_2, the only transition of L, are 2 and 11. The corresponding markings are $Mark([2]) = \{p_2, p_4\}$ and $Mark([11]) = \{p_4, p_7\}$. So $CP = \{\ \{p_2, p_4\}, \{p_4, p_7\}\ \}$.

6.2 The tableau system

Let $\{M_1, \ldots, M_n\}$ be the set of checkpoints obtained in the first phase. Let N_{inv} be the subnet of N obtained by removing from N all visible transitions together with their incident arcs. We will use $\Sigma_1, \ldots, \Sigma_n$ to denote the net systems $(N_{inv}, M_1), \ldots, (N_{inv}, M_n)$.

Definition 8 *Let $BP_1, \ldots, BP_n$ be branching processes of $\Sigma_1, \ldots, \Sigma_n$, respectively. An event e of BP_i is a* repeat *(with respect to $\prec$) if there is an index $j \leq i$ and an event e' in BP_j, called the* companion *of e, such that $Mark([e']) = Mark([e])$, and either*

(I) $j < i$, or

(II) $i = j$ *and* $e' < e$, or

(III) $i = j$, $\neg(e' < e)$, $[e'] \prec [e]$, *and* $||[e']|| \geq ||[e]||$.

A repeat e of BP_i is a terminal *if BP_i contains no repeat $e' < e$. Repeats, and in particular terminals, are of type I, II, or III, according to the condition they satisfy. A repeat e with companion e' is* successful *if it is of type II, and* unsuccessful *otherwise.*

A tableau *is a tuple $BP_1, \ldots, BP_n$ of branching processes of $\Sigma_1, \ldots, \Sigma_n$ such that for every $1 \leq i \leq n$ and for every possible extension e of BP_i at least one of the immediate causal predecessors of e is a terminal. Each BP_i is called a* tableau component*. A tableau is* successful *if at least one of its terminals is successful.*

Observe that an event of BP_i can be a repeat because of an event that belongs to *another* branching process BP_j. The definition of repeat depends on the order of the checkpoints, but the tableau system defined above is sound and complete for any fixed order. Because the definition of the tableau component BP_i depends only on the components with a smaller index, we can create the tableau components in increasing i order. Tableau components are constructed as for the illegal ω-trace problem, using the new definition of terminal.

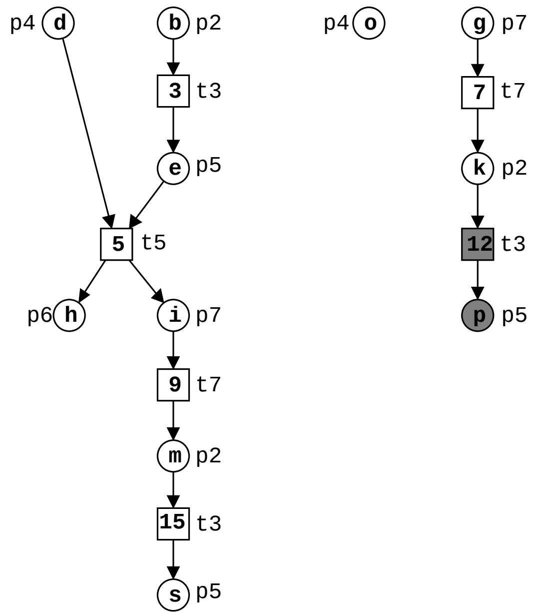

Figure 6: The tableau system for the illegal livelock problem

The tableau for our example is shown in Fig. 6. The names of places and transitions have been chosen to match "pieces" of the unfolding in Fig. 2.

The first tableau component contains no terminals; the construction terminates because no event labelled by an invisible transition can be added. In the second component, event 12 is a terminal with event 3 in the first component as companion. [5] The intuition is that we don't need to unfold beyond 12 in the second component, because what we construct can be found after 3 in the first component.

Similarly to the case of the illegal ω-trace problem, a terminal of type II corresponds to a cycle in the reachability graph. Since the transitions of N_{inv} are all invisible, such a cycle always originates an illegal livelock, and so terminals of type II are always successful. For terminals of type III, the intuition is that $\uparrow[e]$ need not be constructed, because it is isomorphic to $\uparrow[e']$. However, similarly to the case of the tableau system for illegal ω-traces, this doesn't explain why the condition $||[e']|| \geq ||[e]||$ is needed. We can use a slight modification of the net system of Fig. 4 to show that after removing this condition the tableau system is no longer complete. Add a new place p_0, and a new transition t_0 having p_0 as input place, and p_1 as output place. Let $\{p_0\}$ be the initial marking. Let $V = \emptyset$ and $L = \{t_0\}$. The new net system has an illegal livelock, for instance the occurrence sequence $\{p_0\} \xrightarrow{t_0} \{p_1\} \xrightarrow{t_1(t_3t_4)^\omega}$. The set CP contains only one element, namely the marking $\{p_1\}$. So we have one single tableau component $\mathcal{T}_1$. If the condition $||[e']|| \geq ||[e]||$ is removed, then $\mathcal{T}_1$ is identical to the tableau of Fig. 5. Event 3 is a terminal with event 2 as companion, and event 4 is a terminal with event 1 as companion. Both terminals are unsuccessful.

We have the following result (see Appendix A.3):

Theorem 9 *Let $\mathcal{T}_1, \ldots, \mathcal{T}_n$ be a tableau of $\Sigma_1, \ldots, \Sigma_n$ constructed according to a total adequate order $\prec$.*

- *$\mathcal{T}_1, \ldots, \mathcal{T}_n$ is successful if and only if Σ contains an illegal livelock.*
- *$\mathcal{T}_1, \ldots, \mathcal{T}_n$ contain together at most $K^2 \cdot B$ non-terminal events.*

6.3 A tableau system for the 1-safe case

If Σ is 1-safe then we can modify the tableau system to obtain a bound of K^2 non-terminal events. We modify the definition of the repeats of type II and III:

(II') $i = j$ and $\neg(e' \# e)$, or

(III') $i = j$, $e' \# e$, $[e'] \prec [e]$, and $||[e']|| \geq ||[e]||$.

We have the following result (see Appendix A.4):

Theorem 10 *Let Σ be 1-safe. Let $\mathcal{T}_1, \ldots, \mathcal{T}_n$ be a tableau of $\Sigma_1, \ldots, \Sigma_n$ constructed according to a total adequate order $\prec$, and to the new definition of repeats of type II and III.*

- *$\mathcal{T}_1, \ldots, \mathcal{T}_n$ is successful if and only if Σ contains an illegal livelock.*
- *$\mathcal{T}_1, \ldots, \mathcal{T}_n$ contain together at most K^2 non-terminal events.*

---

[5] With a slightly more refined definition of terminal we could have event 7 as terminal, whose companion would be a virtual event having d and b as output conditions.

7 A TABLEAU SYSTEM FOR LTL MODEL CHECKING

Putting the tableau systems of Sections 5 and 6 together, we obtain a tableau system for the model checking problem of $tLTL'$. For the sake of clarity we have considered the illegal ω-trace problem and the illegal livelock problem separately. However, when implementing the tableau systems there is no reason to do so. Since all the branching processes we need to construct are "embedded" in the unfolding of $\Sigma_{\neg\varphi}$, it suffices in fact to construct *one single branching process*, namely the union of all the processes needed to solve both problems.

In our running example we have to construct the union of the nets shown in Fig. 3 (which is both the tableau system for the illegal ω-trace problem and the complete prefix needed to construct the set of checkpoints) and Fig. 6. For each event we have to keep track of the prefixes it belongs to. So, for instance, event 2 belongs to the tableau for the illegal ω-trace problem and to the complete prefix but not to the tableau components for the illegal livelock problem. (The complete prefix will always be a included in the ω-trace tableau, which follows directly from the weaker definition of terminals for the ω-trace tableau.)

Clearly, this prefix contains $\mathcal{O}(K^2 \cdot B)$ non-terminal events. If the system is presented as a 1-safe Petri net, then the prefix contains $\mathcal{O}(K^2)$ non-terminal events because the following two conditions hold:

- By Prop. 1 none of the reachable markings of the synchronization $\Sigma_{\neg\phi}$ enable two I-transitions concurrently. By Theorem 6, the tableau for the illegal ω-trace problem has at most K^2 non-terminal events.

- If the system is a 1-safe Petri net, then the synchronization $\Sigma_{\neg\phi}$ is also 1-safe. By Theorem 9, the tableau for the illegal livelock problem has at most K^2 non-terminal events.

(Note that when we fix the property φ, we obtain the same bounds in the number of reachable markings of the net system Σ, which was given as input to the model checker.)

With small modifications the approach can also handle state based stuttering invariant logics such as LTL-X (the propositional linear temporal logic without the next-time operator). (See Appendix A.1.2.)

8 CONCLUSIONS

We have presented a new unfolding technique for checking LTL-properties. We first make use of the automata-theoretic approach to model checking: a combined system is constructed as the product of the system itself and of an automaton for the negation of the property to be checked. The model checking problem reduces to the illegal ω-trace problem and to the illegal livelock problem for the combined system. Both problems are solved by constructing certain prefixes of the net unfolding of the combined system. In fact, it suffices to construct the union of these prefixes.

The prefixes can be seen as tableau systems for the illegal ω-trace and the illegal livelock problem. We have proved soundness and completeness of these tableau systems, and we have given an upper bound on the size of the tableau. For systems presented as 1-safe Petri nets or products of automata, tableaux contain at most size $\mathcal{O}(K^2)$ (non-terminal) events, where K is the number of reachable states of the system. An interesting open problem is the existence of a better tableau system such that tableaux contain at most $\mathcal{O}(K)$ events. We conjecture that it doesn't exist.

The main advantage of our approach is its simplicity. Wallner's approach proceeds in two steps: construction of a complete prefix, and then construction of a graph. The definition of a graph is non-trivial, and the graph itself can be exponential in the size of the complete prefix. Our approach makes the construction of the graph unnecessary. The price to pay is a larger prefix.

We have solved the illegal ω-trace and the illegal livelock problem in a very general setting. For instance, we allow visible transitions to be concurrent. This may open the way to solutions of the model checking problem for partial order logics interpreted on Mazurkiewicz traces (see for instance [17]). In some of these logics the set of traces violating a property can be characterized as the set of traces of a Petri net. We could then construct the product of the system and this Petri net, and proceed as in this work. This is, however, left for future research.

9 ACKNOWLEDGEMENTS

This work has been partially supported by the Teilprojekt A3 SAM of the Sonderforschungsbereich 342 “Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen”, the Academy of Finland (Project 47754), and the Nokia Foundation.

References

[1] Bibliography on the net unfolding method. Available on the Internet at `http://wwwbrauer.in.tum.de/gruppen/theorie/pom/pom.shtml`.

[2] J. C. Bradfield. *Verifying Temporal Properties of Systems*, volume ISBN 0-8176-3625-0. Birkhäuser, Boston, Massachussets, 1991 (Also: Ph. D. Thesis, University of Edinburgh).

[3] J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23:151–195, 1994.

[4] J. Esparza and S. Römer. An unfolding algorithm for synchronous products of transition systems. In *Proceedings of the 10th International Conference on Concurrency Theory (Concur'99)*, pages 2–20, 1999. LNCS 1055.

[5] J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. In *Proceedings of 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, pages 87–106, 1996. LNCS 1055.

[6] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of 15th Workshop Protocol Specification, Testing, and Verification*, pages 3–18, 1995.

[7] B. Graves. Computing reachability properties hidden in finite net unfoldings. In *Proceedings of 17th Foundations of Software Technology and Theoretical Computer Science Conference*, pages 327–341, 1997. LNCS 1346.

[8] K. Heljanko. Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets. In *Proceedings of 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, pages 240–254, 1999. LNCS 1579.

[9] R. Kaivola. *Equivalences, Preorders and Compositional Verification for Linear Time Temporal Logic and Concurrent Systems.* PhD thesis, University of Helsinki, Department of Computer Science, 1996. 185 p.

[10] R. Kaivola. Using compositional preorders in the verification of sliding window protocol. In *Proceeding of 9th International Conference on Computer Aided Verification (CAV'97)*, pages 48–59, 1997. LNCS 1254.

[11] R. Langerak and E. Brinksma. A complete finite prefix for process algebra. In *Proceeding of 11th International Conference on Computer Aided Verification (CAV'99)*, pages 184–195, 1999. LNCS 1663.

[12] K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Proceedings of 4th Workshop on Computer-Aided Verification (CAV'92)*, pages 164–174, 1992. LNCS 663.

[13] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

[14] S. Melzer and S. Römer. Deadlock checking using net unfoldings. In *Proceedings of 9th International Conference on Computer-Aided Verification (CAV '97)*, pages 352–363, 1997. LNCS 1254.

[15] P. Niebert. Personal communication, 1999.

[16] C. Stirling and David Walker. Local Model Checking in the Modal Mu-calculus. *Theoretical Computer Science*, 89(1):161–177, 1991.

[17] P. S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pages 183–194. IEEE Computer Society Press, 1997.

[18] A. Valmari. On-the-fly verification with stubborn sets. In *Proceeding of 5th International Conference on Computer Aided Verification (CAV'93)*, pages 397–408, 1993. LNCS 697.

[19] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, pages 238–265, 1996. LNCS 1043.

[20] F. Wallner. Model checking techniques using net unfoldings. PhD thesis, Technische Universität München, Germany, forthcoming.

[21] F. Wallner. Model checking LTL using net unfoldings. In *Proceeding of 10th International Conference on Computer Aided Verification (CAV'98)*, pages 207–218, 1998. LNCS 1427.

[22] P. Wolper. Temporal Logic can be more expressive. *Information and Control*, 56(1,2):72–93, 1983.

10 APPENDIXES

APPENDIX A.1 THE SYNCHRONIZATION

Proof of Proposition 1.
None the reachable markings of $\Sigma_{\neg\varphi}$ enable two I-transitions concurrently, and none of the reachable markings of $\Sigma_{\neg\varphi}$ enable a L-transition and a V-transition concurrently.

Proof:
Let M be a reachable marking of $\Sigma_{\neg\varphi}$. The construction of $\Sigma_{\neg\varphi}$ guarantees that $M(q) = 1$ for some Büchi state q, and that and that $M(q') = 0$ for all other Büchi states. (This can be checked by simple marking invariant analysis.) (1) Assume that two I-transitions are concurrently enabled. Because all I- transitions have a Büchi place in their preset, the places corresponding to Büchi states need to have at least two tokens, a contradiction. (2) Assume that a transition $l \in L$ and a transition $v \in V$ such that $l \neq v$ are concurrently enabled. Because l has a Büchi place in preset and also v has a Büchi place in preset, the places corresponding to Büchi states need to have at least two tokens, a contradiction. $\square$

A.1.1 Proof of Theorem 2

First we present the automata theoretic approach to model checking in more detail. Then we show how emptiness checking for the created product automaton can be reduced to the problems of illegal ω-trace and illegal livelock for the synchronization.

Given a labelled net system $\Sigma = (P, T, F, l, M_0)$, and Büchi-automaton $\mathcal{A}_{\neg\varphi} = (\Gamma, Q, q^0, \rho, F)$ we will next define a product Büchi-automaton as $\mathcal{P} = (A, Q', r^0, \rho', F')$, where $r^0 = (M_0, q^0)$, and Q' and ρ' are defined inductively as follows:

- $(M_0, q^0) \in Q'$,
- If $(M, q) \in Q'$, $M \xrightarrow{a} M'$ for some $a \in A$, and $q \xrightarrow{v(a)} q' \in \rho$ then $(M', q') \in Q'$ and $(M, q) \xrightarrow{a} (M', q') \in \rho'$, and
- nothing else is in Q' and ρ'.

The acceptance set is defined to be $F' = \{(M, q) \in Q' \mid q \in F\}$.

The runs of $\mathcal{P}$ correspond to infinite executions of the net system Σ which do not satisfy the formula φ. The language of the automaton $\mathcal{P}$ will thus be non-empty if and only if $\Sigma \not\models \varphi$. We denote by $|RG|$ the number of reachable markings of Σ, and by $|Q|$ the number of states of $\mathcal{A}_{\neg\varphi}$. The number of states of $\mathcal{P}$ is bounded by $|RG| \cdot |Q|$.

We first show that the number of reachable markings of the synchronization as defined in Section 3 is also bounded by the same number of states.

Theorem 11 *The synchronization $\Sigma_{\neg\varphi}$ has at most $(|RG| \cdot |Q|) + 1$ reachable markings.*

Proof:
It follows directly from the construction of $\Sigma_{\neg\varphi}$ that in the initial state the synchronization can only execute the transition $(\hat{p}, \hat{q}) \xrightarrow{\hat{a}} (M_0, q^0)$. After this is done, the construction ensures that all other reachable markings can be represented as pairs (M, q), where M is a reachable marking of Σ and q is a state of $\mathcal{A}_{\neg\varphi}$. $\square$

We formalize the stuttering immunity of $tLTL'$ with the following proposition, see [10, 9].

Proposition 12 *Given a $tLTL'$ formula φ, and two sequences $w, w' \in A^\omega$, if w' can be obtained from w by adding and/or removing invisible transition labels, then it holds that $w' \models \varphi$ if and only if $w \models \varphi$.*

Note that the proposition also holds in some sense for the Büchi-automaton $\mathcal{A}_{\neg\varphi}$. It will accept the image of the word w' under v if and only if it accepts image of the word w under v.

The following lemma shows the correspondence between the emptiness checking of the product automaton and the illegal ω-trace and illegal livelock problems of the synchronization. We define the set $\tau\text{–}accepts(\mathcal{A}_{\neg\varphi})$ as $\{q \in Q \mid \mathcal{A}^q_{\neg\varphi} = (\Gamma, Q, q, \rho, F) \text{ accepts } \tau^\omega\}$, i.e. as the set of states of $\mathcal{A}_{\neg\varphi}$ which will accept the infinite word containing only τ-labels.[6]

Lemma 13 *Let Σ be a labelled net system, $\mathcal{A}_{\neg\varphi}$ be a Büchi-automaton created from a $tLTL'$-formula φ, and $\mathcal{P}$ the product automaton of Σ and $\mathcal{A}_{\neg\varphi}$.*

(1) If the product automaton $\mathcal{P}$ is non-empty then $\Sigma_{\neg\varphi}$ will have either an illegal ω-trace or an illegal livelock.

(2) If $\Sigma_{\neg\varphi}$ has an illegal ω-trace, then the product automaton $\mathcal{P}$ is non-empty.

(3) If $\Sigma_{\neg\varphi}$ has an illegal livelock, then the product automaton $\mathcal{P}$ is non-empty.

Proof:

(1) Let $w = x_0\, x_1\, \ldots \in A^\omega$ be a word accepted by $\mathcal{P}$, and let $\sigma = t_0\, t_1\, \ldots$ be a transition sequence of Σ producing the word w. We will split the proof in two parts: (a) $x_i \in V$ for infinitely many indexes $i \geq 0$, and (b) $x_i \in V$ for only finitely many indexes $i \geq 0$.

(a) Let $w' \in A^\omega$ be the projection of w onto V. By Proposition 12 the logic $tLTL'$ is immune to stuttering of invisible transitions, and thus also the automaton $\mathcal{A}_{\neg\varphi}$ will have an accepting run $r' = q_0\, q_1\, \ldots$ on w'. Given σ and r' we can now create an illegal ω-trace of $\Sigma_{\neg\varphi}$ as follows. First fire the initialization transition $\hat{t}$.

[6] Computing this set can be done in linear time in the size of $\mathcal{A}_{\neg\varphi}$ by e.g. creating the graph of the transition structure of $\mathcal{A}_{\neg\varphi}$ restricted to τ-transitions, computing the non-trivial maximal strongly connected components containing accepting states, and then checking from which states these "accepting" components can be reached.

Then let $j = 0$ and for each index $i \geq 0$ do the following: (i) if $t_i \notin V$ then fire the invisible transition t_i, or (ii) if $t_i \in V$ then fire the transition in $\Sigma_{\neg\varphi}$ corresponding to the synchronization of t_i and the transition $q_j \xrightarrow{l(t_i)} q_{j+1}$ of $\mathcal{A}_{\neg\varphi}$, and let $j = j + 1$. Because w' is the projection of w on V, this sequence of transitions can be fired. Now because some accepting state is repeated infinitely often in r', the fired sequence of transitions will contain infinitely many transitions in I, and will therefore be an illegal ω-trace.

(b) Let $w' \in A^*$ be the projection of w onto V. By Proposition 12 the automaton $\mathcal{A}_{\neg\varphi}$ will have an accepting run r' on $w'\tau^\omega$. Now let q' be the state $\mathcal{A}_{\neg\varphi}$ is in the run r' after reading w'. It follows directly from the definition that $q' \in \tau\!-\!accepts(\mathcal{A}_{\neg\varphi})$. Given σ and r' we can now create an illegal livelock of $\Sigma_{\neg\varphi}$ as follows. First consume all transitions of σ and r' up-to and including the last visible transition exactly as in the case (a). After firing this sequence, the synchronization $\Sigma_{\neg\varphi}$ will have a marking (M, q') for some reachable marking M of Σ, and the last fired transition will be a L-transition ($\hat{t}$ if w contains no visible transitions). Let σ' be the suffix of σ which was not yet fired. It contains an infinite sequence of invisible transitions of Σ enabled in the marking M, which implies that the same sequence of transitions is also enabled at the marking (M, q') of $\Sigma_{\neg\varphi}$. Thus the synchronization will have an illegal livelock.

(2) Let $\sigma = \hat{t}\, t_0\, t_1 \ldots$ be an illegal ω-trace of $\Sigma_{\neg\varphi}$, and let $w = x_0\, x_1 \ldots \in A^\omega$ be the labelling of $t_0\, t_1 \ldots$. If we remove from the sequence σ all invisible transitions, and project the remaining transitions on the Büchi component of the target marking of the corresponding transition, we will obtain an accepting run $r' = q_0\, q_1 \ldots \in Q^\omega$ of $\mathcal{A}_{\neg\varphi}$ on the word $w' \in A^\omega$, such that w' is the projection of w onto V. Thus by Proposition 12, the automaton $\mathcal{A}_{\neg\varphi}$ will also have an accepting run r on the image of w under v. Therefore by removing $\hat{t}$ and projecting the transition sequence σ on the transitions of Σ, we will have a run $t'_0\, t'_1 \ldots$ of Σ and an accepting run r of $\mathcal{A}_{\neg\varphi}$ on the image of the labelling w of $t'_0\, t'_1 \ldots$, which implies $\mathcal{P}$ is non-empty.

(3) Let $\sigma = \hat{t}\, t_0\, t_1 \ldots$ be an illegal livelock of $\Sigma_{\neg\varphi}$, and let $w = x_0\, x_1 \ldots$ be the labelling of the sequence $t_0\, t_1 \ldots$. If we remove from the sequence σ all invisible transitions, and project the remaining transitions on the Büchi component of the target marking of the corresponding transition, we will obtain a sequence $r' = q_0\, q_1 \ldots q' \in Q^*$, such that $q' \in \tau\!-\!accepts(\mathcal{A}_{\neg\varphi})$. Let $r'' \in Q^\omega$ be an accepting run of $\mathcal{A}_{\neg\varphi}$ which begins with r' and after this accepts the suffix τ^ω, and let w' be the word accepted by the run r''. Because w' can be obtained from w by removing invisible transition labels, by Proposition 12 the automaton $\mathcal{A}_{\neg\varphi}$ will also have an accepting run r on the image of w under v. Therefore by removing $\hat{t}$ and projecting the transition sequence σ on the transitions of Σ, we will have a run $t'_0\, t'_1 \ldots$ of Σ and an accepting

run r of $\mathcal{A}_{\neg\varphi}$ on the image of the labelling w of $t'_0\, t'_1\, \ldots$, which implies $\mathcal{P}$ is non-empty.

$\square$

Proof of Theorem 2.
$\Sigma \models \varphi$ *if and only if* $\Sigma_{\neg\varphi}$ *has no illegal* ω*-traces and no illegal livelocks.*

Proof:
The Theorem 2 now directly follows from Lemma 13 and the fact that $\mathcal{P}$ is non-empty if and only if $\Sigma \not\models \varphi$. $\square$

A.1.2 A sketch on handling LTL-X

To handle the state based temporal logic LTL-X, we change the Büchi automaton construction to use e.g. [6], and the synchronization construction used to be the *observation construction* presented in e.g. [21], with the modifications needed to accommodate these changes. The main difference to the action based case is that the set L of $\Sigma_{\neg\psi}$ will need to depend both on the marking of the net system Σ and an the state of the Büchi automaton $\mathcal{A}_{\neg\psi}$. Thus the set L will need to consist of transition instances rather than structural transitions. Given a transition instance of $\Sigma_{\neg\psi}$ corresponding to a move of the Büchi automaton, $(M', q') \xrightarrow{a} (M, q)$ will be in L if and only if $\mathcal{A}^{q}_{\neg\psi} = (\Gamma, Q, q, \rho, F)$ accepts $\mathit{Valuation}(M)^{\omega}$, i.e. if and only if the state q of $\mathcal{A}_{\neg\psi}$ will accept the word consisting of the infinite stuttering of the valuation of atomic propositions in the state M. The computation of the L-transitions instances needs to be done for each visible event added to the livelock tableau. This should not be a major problem because the size of $\mathcal{A}_{\neg\psi}$ is usually quite small and the algorithm needed to run for each transition instance is linear in the size of $\mathcal{A}_{\neg\psi}$.

APPENDIX A.2 PROOFS FOR THE ILLEGAL ω-TRACE TABLEAU

A.2.1 Finiteness

First we define the notion of a cut. A set of conditions is a *co*-set if all its elements are pairwise in *co* relation. A *cut* is a *co*-set of conditions that is maximal with respect to set inclusion. It is easy to prove that the reachable markings of an occurrence net coincide with its cuts. We can assign to a configuration C the marking reached by any of the occurrence sequences mentioned above. This marking is a cut, and it is easy to prove that it is equal to $(\mathit{Min} \cup {}^{\bullet}C) \setminus C^{\bullet}$, where Min denotes the set of minimal places of the branching process.

Let K denote the number of reachable markings of Σ, and B denote the maximum number of tokens the reachable markings of Σ put in the places of Σ. We need the following lemma.

Lemma 14 *Let* E *be a finite subset of events of a configuration* C*, and let* $k \geq 0$*. If* E *contains more than* $k \cdot B$ *events, then there exist* $e_1, \ldots, e_{k+1} \in E$ *such that* $e_1 < e_2 < \ldots < e_{k+1}$.

Proof:
We observe first that no *co*-set of conditions of C contains more than B elements: otherwise there exists a cut containing more than B conditions, and since this cut corresponds to a reachable marking, Σ has a reachable marking which puts more than B tokens on the set of places P, contradicting the definition of B. It follows that no *co*-set of events contains more than B elements, because the output conditions of these events also form a *co*-set, and each event has at least one output condition. In particular, no *co*-set of events of E contains more than B elements.

The proof of the lemma is by induction on k.
Basis. $k = 0$. Then there is nothing to prove.
Step. $k > 0$. Let $Max(E)$ be the set of maximal elements of E with respect to the causal relation. Since $Max(E)$ is a *co*-set, it contains at most B elements. Let $E' = E \setminus Max(E)$. E' contains more than $(k-1) \cdot B$ events, and so by induction hypothesis there exist $e_1, \ldots, e_k \in E'$ such that $e_1 < e_2 < \ldots < e_k$. Since e_k does not belong to $Max(E)$, we have $e_k < e_{k+1}$ for some event $e_{k+1} \in Max(E)$. So $e_1 < e_2 < \ldots < e_k < e_{k+1}$, and we are done. □

Proof of Theorem 6 - finiteness.

(a) *$\mathcal{T}$ contains at most $K^2 \cdot B$ non-terminal events.*

(b) *If the transitions of I are pairwise non-concurrent, then $\mathcal{T}$ contains at most K^2 non-terminal events.*

Proof:
(a) We proceed in three steps.

(1) Let C be an configuration of $\mathcal{T}$. If $|C| > K \cdot B$, then C contains a terminal.
If $|C| > K \cdot B$, then, by Lemma 14, C contains a chain $e_1 < \ldots < e_{K+1}$. By the pigeonhole principle, there are events $e_i < e_j$, $1 \leq i, j \leq K+1$, such that $Mark([e_i]) = Mark([e_j])$. So e_j is a repeat. It follows that e_j is either a terminal or a causal successor of a terminal. By the definition of tableau, e_j is a terminal.

(2) Let M be a reachable marking of Σ. $\mathcal{T}$ contains at most $K \cdot B$ non-terminal events e such that $Mark(e) = M$.
Assuming the contrary, let $e_1 \ldots e_m$ be pairwise different non-terminal events such that $m > K \cdot B$, and $Mark(e_i) = M$ for all $1 \leq i \leq m$. By (1), we have $|[e_i]| \leq K \cdot B$ and so $\#_I[e_i] \leq K \cdot B$ for all $1 \leq i \leq m$. By the pigeonhole principle, there are two indices $i \neq j$ such that $\#_I[e_i] = \#_I[e_j]$. Since $\prec$ is a total order, we have either $[e_i] \prec [e_j]$ or $[e_j] \prec [e_i]$. So either e_i or e_j is a repeat, and, by the definition of tableau, a terminal.

(3) $\mathcal{T}$ contains at most $K^2 \cdot B$ non-terminal events.
By (2), $\mathcal{T}$ contains at most $K \cdot B$ non-terminal events for each reachable marking, and so $K^2 \cdot B$ non-terminal events.

(b) We proceed in three steps.

(1) Let C be an configuration of $\mathcal{T}$. If C contains more than K events labelled by transitions of I, then C contains a terminal.
Since all events labelled by transitions of I are causally related, C contains a chain $e_1 < \ldots < e_{K+1}$ of such events. Proceed now as in (a)(1).

(2) Let M be a reachable marking of Σ. $\mathcal{T}$ contains at most K non-terminal events e such that $Mark(e) = M$.
Assume the contrary, and let $e_1 \ldots e_{K+1}$ be pairwise different non-terminal events such that $Mark(e_i) = M$ for all $1 \leq i \leq K+1$. By (1), we have $\#_I[e_i] \leq K$ for all $1 \leq i \leq K+1$. So there are two indices $i \neq j$ such that $\#_I[e_i] = \#_I[e_j]$. Proceed now as in (a)(2).

(3) $\mathcal{T}$ contains at most K^2 non-terminal events.
Proceed as in (a)(3).

□

A.2.2 Soundness

Soundness follows easily from the definition of a successful terminal.

Proof of Theorem 6 - soundness.
If $\mathcal{T}$ is successful then Σ has an illegal ω-trace.

Proof:
If $\mathcal{T}$ is successful, then it contains a successful terminal e with companion e'. In particular, $e' < e$, and so $[e'] \subset [e]$. Let $M_0 \xrightarrow{\sigma} M_1 \xrightarrow{\sigma_1} M_2$ be a firing sequence of Σ such that σ and $\sigma\sigma_1$ are linearisations of $[e']$ and $[e]$, respectively. Since $Mark([e]) = Mark([e'])$, we have $M_1 = M_2$. Since $[e] \setminus [e']$ contains some I-event, $M_0 \xrightarrow{\sigma} M_1 \xrightarrow{\sigma_1^\omega}$ is an infinite firing sequence of Σ containing infinitely many occurrences of transitions of I. □

A.2.3 Completeness

The proof of completeness is a bit more involved. We need a preliminary definition, and a lemma.

Definition 15 *A configuration C of a branching process of Σ is* bad *if it contains at least $(K \cdot B) + 1$ I-events.*

Lemma 16 *(1) Σ has an illegal ω-trace if and only if its unfolding contains a bad configuration.*

(2) A bad configuration contains at least one terminal. (More precisely, if a branching process of Σ contains a bad configuration, then some event of this configuration is a terminal.)

Proof:
(1) ($\Rightarrow$): Let $M_0 \xrightarrow{\sigma} M$ be a prefix of an illegal ω-trace such that σ contains $(K \cdot B) + 1$ occurrences of I-transitions. There exists a configuration of the unfolding of Σ such that σ is a linearisation of C. This configuration is bad.

(1) ($\Leftarrow$): Let C be a bad configuration. C contains at least $(K \cdot B) + 1$ I-events. By Lemma 14, C contains a chain $e_1 < \ldots < e_{K+1}$ of I-events. By the pigeonhole principle, two of these I-events, say e_i, e_j, satisfy $Mark([e_i]) = Mark([e_j])$, and $e_i < e_j$. Now proceed as in the proof of Theorem 6 - soundness.

(2) Event e_j in the proof of (1) ($\Leftarrow$) is a repeat. Since C contains a repeat, it also contains a terminal. □

Proof of Theorem 6 - completeness.
If Σ has an illegal ω-trace then $\mathcal{T}$ is successful.

Proof:
By Lemma 16(1), it suffices to show that if the unfolding of Σ contains a bad configuration, then $\mathcal{T}$ is successful.

We prove that, given a bad configuration C of the unfolding of Σ, either C contains a successful terminal of $\mathcal{T}$ (and so $\mathcal{T}$ itself contains a successful terminal), or there exists another bad configuration C' such that $C' \prec C$. Since $\prec$ is well founded (see Definition 4), $\mathcal{T}$ contains a successful terminal.

If C contains a successful terminal of $\mathcal{T}$, then we are done. Otherwise, by Lemma 16(2), C contains an unsuccessful terminal e. We have $C = [e] \oplus (C \setminus [e])$. Let e' be the companion of e, and let f be an isomorphism between $\uparrow[e]$ and $\uparrow[e']$. Define $C' = [e'] \oplus f(C \setminus [e])$. We prove that C' is a bad configuration satisfying $C' \prec C$.

We consider two cases, corresponding to the two possibilities for a terminal to be unsuccessful:

- $e' < e$, and $[e] \setminus [e']$ contains no I-event.
  In this case we have $[e'] \subseteq [e]$. By the second condition in the definition of an adequate order, we have $[e'] \prec [e]$. By the third condition of the same definition, we have $C' \prec C$.
  It remains to prove that C' is bad. We show that C and C' contain the same number of I-events. Since $[e] \setminus [e']$ contains no I-event, we have $\#_I[e] = \#_I[e']$. Since isomorphisms preserve labelling, we have $\#_I f(C \setminus [e]) = \#_I(C \setminus [e])$. So $\#_I C' = \#_I C$.

- $[e'] \prec [e]$ and $\#_I[e'] \geq \#_I[e]$.
  Since $[e'] \prec [e]$, we have $C' \prec C$ by the third condition in the definition of an adequate order. It remains to prove that C' is bad. We show that C' contains at least as many I-events as C. We have $\#_I[e'] \geq \#_I[e]$ by assumption. Since isomorphisms preserve labelling, we also have $\#_I f(C \setminus [e]) = \#_I(C \setminus [e])$. So $\#_I C' \geq \#_I C$.

□

APPENDIX A.3 PROOFS FOR THE ILLEGAL LIVELOCK TABLEAU

A.3.1 Finiteness

Proof of Theorem 9 - finiteness.
$\mathcal{T}_1, \ldots, \mathcal{T}_n$ contain together at most $K^2 \cdot B$ non-terminal events.

Proof:
The proof follows the pattern of the proof of Theorem 6 - finiteness (a). Parts (1), and (3) are proved exactly as in that theorem. Only part (2) requires a new proof.

(2) Let M be a reachable marking of Σ. $\mathcal{T}_1, \ldots, \mathcal{T}_n$ contain together at most $K \cdot B$ non-terminal events e such that $Mark(e) = M$.
Assuming the contrary, let E be a set containing more than $K \cdot B$ non-terminal events such that $Mark(e) = M$ for all $e \in E$. By (1), we have $||[e]|| \leq K \cdot B$ for all $e \in E$. By the pigeonhole principle, there are two different events e_1, e_2 such that $||[e_1]|| = ||[e_2]||$. We show that e_1 or e_2 is a repeat. There are three possible cases:

- e_1 and e_2 belong to different tableau components.
Then the one belonging to the component with the larger index is a repeat.
- e_1 and e_2 belong to the same tableau component, and are causally related.
Then the largest one with respect to the causal relation is a repeat.
- e_1 and e_2 belong to the same tableau component, and are not causally related.
Then e_1 and e_2 are ordered by $\prec$, because $\prec$ is a total order. Since $||[e_1]|| = ||[e_2]||$, the largest of the two with respect to $\prec$ is a repeat.

By the definition of a tableau, either e_1 or e_2 is a terminal, a contradiction.

□

A.3.2 Soundness

As in the case of the illegal ω-trace problem, soundness follows immediately from the definition of a successful terminal.

Proof of Theorem 9 - soundness.
If $\mathcal{T}_1, \ldots, \mathcal{T}_n$ is successful then Σ contains an illegal livelock.

Proof:
If $\mathcal{T}_1, \ldots, \mathcal{T}_n$ is successful, then it contains a successful terminal e with companion e'. In particular, e and e' belong to the same tableau component, say $\mathcal{T}_i$, and $e' < e$, which implies $[e'] \subset [e]$. Let $M_i \xrightarrow{\sigma} M_1 \xrightarrow{\sigma_1} M_2$ be a firing sequence of (N_{inv}, M_i) such that σ is a linearisation of $[e']$ and $\sigma\sigma_1$ is a linearisation of $[e]$. Since $Mark([e]) = Mark([e'])$, we have $M_1 = M_2$. Since M_i is a checkpoint of Σ, there exists a firing sequence $M_0 \xrightarrow{\sigma'} M_i$ such that the last transition of σ' belongs to L. So $M_0 \xrightarrow{\sigma'} M_i \xrightarrow{\sigma\sigma_1^\omega}$ is an illegal livelock of Σ. □

A.3.3 Completeness

The completeness proof uses the following notion:

Definition 17 *An L-pair is a pair (C, e) where C is a configuration of the unfolding of Σ, and e is an event of C, satisfying the following properties:*

- *e is labelled by a transition of L,*
- *all events of $C \setminus [e]$ are labelled by invisible transitions, and*
- *the set $C \setminus [e]$ contains at least $(K \cdot B) + 1$ events.*

An L-pair (C, e) is minimal *if $[e] \preceq [e']$ for any other L-pair (C', e').*

Loosely speaking, the following lemma shows that L-pairs are finite witnesses of the existence of illegal livelocks. Moreover, minimal L-pairs are related to checkpoints. This lemma makes use of our assumption stating that no reachable marking concurrently enables a transition of V and a transition of L.

Lemma 18 *(1) Σ has an illegal livelock $M_0 \xrightarrow{\sigma} M \xrightarrow{\sigma_1}$ if and only if its unfolding contains an L-pair (C, e) such that $Mark([e]) = M$.*

(2) If (C, e) is a minimal L-pair, then $Mark([e])$ is a checkpoint of Σ.

Proof:
(1)($\Rightarrow$): Let $M_0 \xrightarrow{\sigma} M \xrightarrow{\sigma_1}$ be a livelock of Σ, i.e.,

- the last transition of σ belongs to L, and
- σ_1 is an infinite sequence containing only invisible transitions.

Let C be an (infinite) configuration of the unfolding of Σ such that $\sigma\sigma_1$ is one of its linearisations. C contains an event e corresponding to the last transition of σ. Since C is infinite, $C \setminus [e]$ contains infinitely many events. Let $[e] \oplus E$ be an extension of $[e]$ such that E contains at least $(K \cdot B) + 1$ events. The events of E are either concurrent to or causal successors of e. In the latter case, they are labelled with transitions of σ_1, and so invisible. In the former case, since no reachable marking concurrently enables the last transition of σ and a visible transition, none of these events is labelled by a visible transition. So all events of E are labelled by invisible transitions, which implies that $([e] \oplus E, e)$ is an L-pair.

(1)($\Leftarrow$): Let (C, e) be an L-pair, and let $M = Mark([e])$. Let $M_0 \xrightarrow{\sigma} M$ be a linearisation of $[e]$. Since e is the unique maximal event of $[e]$, it is labelled by the last transition of σ; since (C, e) is an L-pair, e is labelled by a transition of L, and so the last transition of σ belongs to L. We will now construct an infinite firing sequence $M \xrightarrow{\sigma_2\sigma_3^\omega}$ of invisible transitions, which implies that $M_0 \xrightarrow{\sigma} M \xrightarrow{\sigma_2\sigma_3^\omega}$ is an illegal livelock.

Since (C, e) is an L-pair, the events of $C \setminus [e]$ are labelled by invisible transitions, and so the unfolding of (N_{inv}, M) contains a configuration C' isomorphic to $C \setminus [e]$. Since $C \setminus [e]$ contains at least $(K \cdot B) + 1$ events, so does C'. By Lemma 14, C' contains a chain $e_1 < \ldots < e_{K+1}$. By the pigeonhole principle, there are events $e_i < e_j$ such that $Mark(e_i) = Mark(e_j)$. Let $M \xrightarrow{\sigma_2} M_1 \xrightarrow{\sigma_3} M_2$ be a linearisation of $[e_j]$ such that $M \xrightarrow{\sigma_2} M_1$ is a linearisation of $[e_i]$. We have $M_1 = M_2$, and so $M \xrightarrow{\sigma_2} M_1 \xrightarrow{\sigma_3^\omega}$ is an infinite firing sequence. Since the transitions of $C \setminus [e]$ are labelled by invisible actions, $\sigma_2 \sigma_3^\omega$ contains only invisible transitions, and we are done.

(2) Since (C, e) is an L-pair, e is labelled by a transition of L, and so it suffices to show that e belongs to the finite prefix of Σ. Assume this is not the case. Then, some event $d < e$ is a terminal of the complete finite prefix of Σ. Let d' be the companion of d. Fix an isomorphism f between $\uparrow [d']$ and $\uparrow [d]$, and let $C' = f(C)$ and $e' = f(e)$. Since f preserves labelling, e' is labelled by a transition of L, and all events of $C' \setminus [e']$ are labelled by invisible transitions. Moreover, $C' \setminus [e']$ contains exactly as many events as $C \setminus [e]$, and so at least $(K \cdot B) + 1$ events. It follows that (C', e') is an L-pair. We show $[e'] \prec [e]$, which contradicts the minimality of (C, e). By the definition of a terminal, we have $[d'] \prec [d]$. Since $d < e$, we have $[e] = [d] \oplus D$ for some set of events D. It follows $[e'] = [d'] \oplus f(D)$. By the definition of an adequate order, $[e'] \prec [e]$, and we are done. □

Loosely speaking, our next lemma shows that minimal L-pairs lead to terminals in the tableau system. Before we can formulate it we need some notations. Let $CP = \{M_1, \ldots, M_n\}$, and fix a minimal L-pair (C, e). By Lemma 18(2), $\mathit{Mark}([e])$ is an element of CP, say M_i. It follows that $\uparrow [e]$ restricted to the invisible events, and the unfolding of Σ_i are isomorphic. Fix an isomorphism f_i from $\uparrow [e]$ restricted to the invisible events to the unfolding of Σ_i, and define $C_e = f_i(C \setminus [e])$. Clearly, C_e is a configuration of the unfolding of Σ_i.

Lemma 19 *The configuration C_e contains a terminal. (More precisely, in any tuple of branching processes $BP_1, \ldots, BP_n$ of $\Sigma_1, \ldots, \Sigma_n$, if BP_i contains C_e, then some event of C_e is a terminal.)*

Proof:
Since (C, e) is an L-pair, the set $C \setminus [e]$ contains at least $(K \cdot B) + 1$ events. Since C_e is isomorphic to $C \setminus [e]$, it also contains at least $(K \cdot B) + 1$ events. By Lemma 14, C_e contains a chain $e_1 < \ldots < e_{K+1}$. By the pigeonhole principle there are events $e_i < e_j$ such that $\mathit{Mark}([e_i]) = \mathit{Mark}([e_j])$. So e_j is a repeat, and C_e contains a terminal. □

Proof of Theorem 9 - completeness.
If Σ contains an illegal livelock then $\mathcal{T}_1, \ldots, \mathcal{T}_n$ is successful.

Proof:
By Lemma 18(1) Σ has an L-pair, and so it also has a minimal L-pair (C, e). By Lemma 18(2), Σ has an illegal livelock $M_0 \xrightarrow{\sigma} M_i \xrightarrow{\sigma_1}$ such that $M_i = \mathit{Mark}([e])$, and M_i is a checkpoint. Without loss of generality, we choose (C, e) so that the index i is minimal.

We prove that $\mathcal{T}_i$ contains a successful terminal. We proceed as follows: We show that either the configuration C_e of the unfolding of Σ_i contains a successful terminal of $\mathcal{T}_i$, or there exists another minimal L-pair (C', e) (observe that the event e doesn't change) such that $C'_e \prec C_e$. Since $\prec$ is well founded, $\mathcal{T}_i$ must contain a successful terminal.

If C_e contains a successful terminal of $\mathcal{T}_i$, then we are done. Otherwise, by Lemma 19, it contains an unsuccessful terminal d. Let d' be the companion of d.

We first prove that d' is also an event of $\mathcal{T}_i$. Assume the contrary. Then, by the definition of a terminal, d' is an event of $\mathcal{T}_j$ and $j < i$. Since d is an event of $\mathcal{T}_i$, we can split σ_1 into two sequences, $\sigma_1 = \sigma_2\sigma_3$, such that σ_2 is a linearisation of $[d]$, and so

$$M_0 \xrightarrow{\sigma} M_i \xrightarrow{\sigma_2} \mathit{Mark}([d]) \xrightarrow{\sigma_3}$$

Since $\mathit{Mark}([d']) = \mathit{Mark}([d])$, we find a linearisation σ_2' of $[d']$ in $\mathcal{T}_j$ such that
$M_j \xrightarrow{\sigma_2'} \mathit{Mark}([d']) = \mathit{Mark}([d])$. So we have

$$M_0 \xrightarrow{\sigma'} M_j \xrightarrow{\sigma_2'} \mathit{Mark}([d']) = \mathit{Mark}([d]) \xrightarrow{\sigma_3}$$

which is an illegal livelock of Σ. Since M_j is a checkpoint and $j < i$, we reach a contradiction to our assumption that the index i is minimal.

Since d' is also an event of $\mathcal{T}_i$, and d is an unsuccessful terminal, we have $[d'] \prec [d]$, and $|[d']| \geq |[d]|$. We construct a configuration C' of the unfolding of Σ in two steps.

- Define $C_e' = [d'] \oplus f(C_e \setminus [d])$, where f is an isomorphism from $\uparrow[d]$ to $\uparrow[d']$.

- Define $C' = [e] \oplus f_i^{-1}(C_e')$, where f_i is the isomorphism from $\uparrow[e]$ to the unfolding of Σ_i we used above.

We make the following observations:

- C_e' is a configuration of $\mathcal{T}_i$.
  Follows immediately from the definitions.

- All events of C_e' are labelled by invisible transitions.
  Because $\mathcal{T}_i$ is a branching process of $\Sigma_i = (N_{inv}, M_i)$, and N_{inv} contains only invisible transitions.

- C_e' contains at least $(K \cdot B) + 1$ events.
  Since $|[d']| \geq |[d]|$, we have $|C_e'| \geq |C_e|$ by the definition of C_e'. Since (C, e) is an L-pair, C_e contains at least $(K \cdot B) + 1$ events, and so C_e' also contains at least $(K \cdot B) + 1$ events.

It follows that (C', e) is a minimal L-pair. We now show $C_e' \prec C_e$. Since $[d'] \prec [d]$ by assumption, and $\prec$ is an adequate order, we have

$$\begin{array}{rcll} C_e' & = & [d'] \oplus f(C_e \setminus [d]) & \text{(definition of } C_e'\text{)} \\ & \prec & [d] \oplus (C_e \setminus [d]) & ([d'] \prec [d] \text{ and } \prec \text{ is an adequate order)} \\ & = & C_e & \end{array}$$

□

APPENDIX A.4 THE 1-SAFE CASE

The new tableau system has weaker requirements for an event to be a repeat, and so it is complete. More precisely, a repeat of type II is also a repeat of type II′, and a repeat of type III is either a repeat of type II′ (if $\neg(e\#e')$) or a repeat of type III′ (if $e\#e'$). We deal with soundness, and with the new size bound.

A.4.1 Finiteness

Proof of Theorem 10 - finiteness.
$\mathcal{T}_1, \ldots, \mathcal{T}_n$ contain together at most K^2 non-terminal events.

Proof:
The proof has the same parts as that of Theorem 6 - finiteness (a). Parts (2) and (3) are proved as in that theorem, only (1) requires a new proof.

(1) Let C be an configuration of $\mathcal{T}_1, \ldots, \mathcal{T}_n$. If $|C| > K$, then C contains a terminal.
If $|C| > K$, then C contains two events e, e' such that $\mathit{Mark}([e]) = \mathit{Mark}([e'])$. By the definition of repeat of type II', e or e' is a repeat. By the definition of tableau, C contains a terminal.

□

A.4.2 Soundness

For soundness, we need the following lemma:

Lemma 20 *Let C_1, C_2 configurations of the unfolding of a 1-safe net system Σ such that $C_1 \cup C_2$ is a configuration, and $\mathit{Mark}(C_1) = M = \mathit{Mark}(C_2)$. Then $\mathit{Mark}(C_1 \cap C_2) = M$.*

Proof:
Let c_1, c_2, c_{12} be the cuts corresponding to C_1, C_2, and $C_1 \cap C_2$, respectively.

(1) If $p \in M$, then $p \in \mathit{Mark}(C_1 \cap C_2)$.

Since $p \in M$, there exist $b_1 \in c_1$, $b_2 \in c_2$ such that $l(b_1) = p = l(b_2)$. If $b_1 = b_2$, then $b_1 \in c_{12}$, and thus $p \in \mathit{Mark}(C_1 \cap C_2)$. Otherwise $b_1 \neq b_2$, and since Σ is 1-safe, b_1 and b_2 cannot be concurrent. Since $C_1 \cup C_2$ is a configuration, they must be causally related. Assume without loss of generality that $b_1 < b_2$. Then $b_1 \in c_{12}$, and so $p \in \mathit{Mark}(C_1 \cap C_2)$.

(2) If $p \in \mathit{Mark}(C_1 \cap C_2)$, then $p \in M$.

Since $p \in \mathit{Mark}(C_1 \cap C_2)$, there exists $b \in c_{12}$ such that $l(b) = p$. We have:

(a) The unique event of ${}^\bullet b$ belongs to $C_1 \cap C_2$.
This follows immediately from $b \in c_{12}$.

(b) $b^\bullet \cap C_1 = \emptyset$ or $b^\bullet \cap C_2 = \emptyset$.
Assume the contrary. Then, there are events $e_1 \in b^\bullet \cap C_1$ and $e_2 \in b^\bullet \cap C_2$. If $e_1 = e_2$, then $e_1 \in C_1 \cap C_2$, which implies $b \notin c_{12}$, a contradiction. If $e_1 \neq e_2$, then e_1 and e_2 are in conflict, and so $C_1 \cup C_2$ is not a configuration, a contradiction.

By (a) and (b) we have $b \in c_1$ or $b \in c_2$, and so $p \in Mark(C_1)$ or $p \in Mark(C_2)$. Since $Mark(C_1) = M = Mark(C_2)$, we have $p \in M$. $\square$

Proof of Theorem 10 - soundness.
If $\mathcal{T}_1, \ldots, \mathcal{T}_n$ is successful then Σ contains an illegal livelock.

Proof:
If $\mathcal{T}_1, \ldots, \mathcal{T}_n$ is successful, then it contains a successful terminal e with companion e'. In particular, e and e' belong to the same tableau component. If $e' < e$, then we proceed as in the proof of Theorem 9 - soundness. If e' *co* e, then there exists a firing sequence $M_i \xrightarrow{\sigma} M_1 \xrightarrow{\sigma_1} M_2$, where σ is a linearisation of $[e'] \cap [e]$ and $\sigma\sigma_1$ is a linearisation of $[e]$. By Lemma 20, $Mark([e'] \cap [e]) = Mark([e])$, and so $M_1 = M_2$. Finally, since M_i is a reachable marking of Σ, there exists a firing sequence $M_0 \xrightarrow{\sigma'} M_i$ such that the last transition of σ' belongs to L. So $M_0 \xrightarrow{\sigma'} M_i \xrightarrow{\sigma{\sigma_1}^{\omega}}$ is an illegal livelock of Σ. $\square$

HELSINKI UNIVERSITY OF TECHNOLOGY LABORATORY FOR THEORETICAL COMPUTER SCIENCE
RESEARCH REPORTS

HUT-TCS-A47	Patrik Simons Towards Constraint Satisfaction through Logic Programs and the Stable Model Semantics. August 1997.
HUT-TCS-A48	Tuomas Aura On the structure of delegation networks. December 1997.
HUT-TCS-A49	Tomi Janhunen Non-Monotonic Systems: A Framework for Analyzing Semantics and Structural Properties of NMR. March 1998.
HUT-TCS-A50	Ilkka Niemelä (Ed.) Proceedings of the HeCSE Workshop on Emerging Technologies in Distributed Systems. March 1998.
HUT-TCS-A51	Kimmo Varpaaniemi On the Stubborn Set Method in Reduced State Space Generation. May 1998.
HUT-TCS-A52	Ilkka Niemelä, Torsten Schaub (Eds.) Proceedings of the Workshop on Computational Aspects of Nonmonotonic Reasoning. May 1998.
HUT-TCS-A53	Stefan Rönn Semantics of Semaphores. 1998.
HUT-TCS-A54	Antti Huima Analysis of Cryptographic Protocols via Symbolic State Space Enumeration. August 1999.
HUT-TCS-A55	Tommi Syrjänen A Rule-Based Formal Model For Software Configuration. December 1999.
HUT-TCS-A56	Keijo Heljanko Deadlock and Reachability Checking with Finite Complete Prefixes. December 1999.
HUT-TCS-A57	Tommi Junttila Detecting and Exploiting Data Type Symmetries of Algebraic System Nets during Reachability Analysis. December 1999.
HUT-TCS-A58	Patrik Simons Extending and Implementing the Stable Model Semantics. April 2000.
HUT-TCS-A59	Tommi Junttila Computational Complexity of the Place/Transition-Net Symmetry Reduction Method. April 2000.
HUT-TCS-A60	Javier Esparza, Keijo Heljanko A New Unfolding Approach to LTL Model Checking. April 2000.

ISBN 951-22-4999-5
ISSN 0783-5396