Improving Automata Generation for Linear Temporal Logic by Considering the Automaton Hierarchy

Klaus Schneider

University of Karlsruhe, Department of Computer Science
Institute for Computer Design and Fault Tolerance
P.O. Box 6980, 76128 Karlsruhe, Germany
email: Klaus.Schneider@informatik.uni-karlsruhe.de
http://goethe.ira.uka.de/∼schneider

Abstract. *We present new algorithms to translate linear time temporal logic (LTL) formulas with past operators to equivalent ω-automata. The resulting automata are given in a symbolic representation that directly supports symbolic model checking. Furthermore, this has the advantage that the translations run in linear time wrt. the length of the input formula. To increase the efficiency of the model checking, our translations avoid as far as possible the introduction of computationally expensive fairness constraints, or at least replace them by simpler reachability constraints. Using the well-known automaton hierarchy, we show that our improvements are complete. Finally, we show how large parts of the formulas can be translated to the simpler logic CTL, which accelerates the LTL model checking by orders of magnitude which is shown by experimental results.*

1 Introduction

The reactive behavior of concurrent systems can be conveniently specified with temporal logics [10] like CTL [5], LTL [24], and CTL* [11]. These logics have a different expressiveness and also very different verification procedures: CTL* model checking can be easily reduced to LTL model checking [12], LTL model checking in turn is reduced to nonemptiness problems of ω-automata, and CTL model checking is reduced to model checking of the alternation-free μ-calculus. While symbolic CTL model checking is very efficient [3], some authors complain about the hard restrictions of CTL [17, 26], and therefore tend to use the more comfortable logics LTL and CTL* [35]. All modern model checking tools therefore support LTL.

Verification procedures for LTL seem, however, not to be as powerful as those for CTL [6, 33]. Usually, the given LTL formulas are translated to equivalent ω-automata whose emptiness is then to be checked. Similar to the verification procedures, there are two different approaches to these translations: procedures that construct the automata explicitly like [19, 34, 23, 15, 9, 31, 13, 38, 14], and others that derive a symbolic description of the automata like [3, 6, 18, 29, 8]. The latter have the advantage that they run with linear runtime and memory requirements wrt. the length of the formulas. Moreover, they can be directly used for symbolic model checking.

All of the mentioned translation procedures construct special ω-automata, usually *generalized Büchi automata* (a special form of Streett automata [32]). The acceptance of these automata is defined by a set of sets of states $\{Q_1, \ldots, Q_n\}$: an infinite input

sequence is accepted by such an automaton iff there is a run through the state transition system that visits each Q_i infinitely often. In the following, the Q_i's are called *fairness constraints* (as usual in CTL model checking). It is well-known that the verification of such fairness constraints requires a nested fixpoint iteration which makes them hard to verify[1], especially in symbolic model checking [25, 1].

In this paper, we reconsider the translation procedures as given in [3, 6, 18, 29], and show that these procedures can be significantly improved. The problem with these procedures is that each temporal future operator (except for the next-time operator) of the considered LTL formula induces a fairness constraint during the translation. It is well-known that the translation of *arbitrary* temporal formulas is not possible without fairness constraints. Nevertheless, there are a lot of specifications, including all *safety and liveness properties* [20], that can be translated to simpler classes of ω-automata (whose acceptance condition does not require fairness constraints). Consequently, some of the fairness constraints introduced by the procedures [3, 6, 18, 29] are unnecessary, and others (still used by [8]) can be replaced with simpler constraints.

We therefore present two powerful improvements of the symbolic translation procedures that still retain the linear runtime of the translation. The first improvement is based on exploiting the monotonicity of logical operators and therefore allows one to *neglect the introduction of those fairness constraints that stem from positive/negative occurrences of weak/strong temporal operators*. This is already used in [8] and in most of the explicit translation procedures, but not in [3, 6, 18, 29]. The second improvement allows one to *replace some of the remaining fairness constraints by simpler reachability constraints*. This improvement can be used as long as only strong temporal future operators are nested into each other, followed by only nestings of weak temporal future operators. To the best of our knowledge such an improvement has not been used so far. Based on these improvements, we define new subclasses of LTL that can be translated without fairness constraints at all, i.e., either without any constraints, or with only reachability constraints.

Beneath the translations of subclasses of LTL to corresponding automaton classes, we also consider a combination with the methods presented in [26, 27]. These procedures allows one to translate large parts of the LTL formulas to CTL, thus bridging the gap between LTL and CTL. We have implemented all translation procedures in a front-end for the SMV model checker family [21, 22, 4], which yields a new verification tool for LTL specifications that outperforms the direct use of these tools. The tool has been completely written in Java, so that it is platform-independent and can be freely accessed under `http://goethe.ira.uka.de/~schneider/my_tools`.

The paper is organized as follows: In the next section, we define the syntax and the semantics of the temporal logic that is used in the paper. After this, we define our *new temporal logics in correspondence to the well-known automaton hierarchy*. In section 4, we review the translation procedures of [3, 6, 29]. In section 5, we present our improvements and hence, the first variant of our new translation procedures. We then combine all our improvements in section 6. We also show there how parts of the LTL formulas

[1] The best known algorithm [7] for checking a μ-calculus formula Φ in a finite state system with $|\mathcal{S}|$ states and $|\mathcal{R}|$ transitions is of order $O((|\mathcal{S}|\,|\Phi|)^{\mathsf{ad}(\Phi)-1}\,|\mathcal{R}|\,|\Phi|)$, where $\mathsf{ad}(\Phi)$ is the alternation depth of Φ. Fairness constraints increase the alternation depth to $\mathsf{ad}(\Phi) = 2$.

can be directly translated to CTL to further increase the efficiency. Experimental results that prove the power of our improvements are finally given in section 7. Due to lack of space, we do not list proofs; however, we note that most parts of the translation procedures have been checked with the HOL [16] proof assistant, and a more detailed description of the algorithms together with proofs can be found in [28].

Some recent work can be combined with the translations presented here: [13] and [31] list different sets of rewrite rules to simplify LTL formulas before translating them to automata. Moreover, [2] presents symbolic procedures for checking nonemptiness of weak and terminal Büchi-automata (which are called $\mathsf{NDet_{FG}}$ and $\mathsf{NDet_{F}}$ below), which yields a perfect back-end for our translations.

2 Syntax and Semantics of $\mathcal{L}_\omega$

We first define an extended temporal logic $\mathcal{L}_\omega$ (as did many authors like [37] before) to present our translations in a single formalism.

Definition 1 (Syntax of $\mathcal{L}_\omega$). *The following mutually recursive definitions introduce the set of $\mathcal{L}_\omega$ formulas over a given finite set of variables $\mathcal{V}$:*

- *each variable is a $\mathcal{L}_\omega$ formula, i.e., $\mathcal{V} \subseteq \mathcal{L}_\omega$*
- *$\neg\varphi$, $\varphi \vee \psi \in \mathcal{L}_\omega$ if $\varphi, \psi \in \mathcal{L}_\omega$*
- *$\mathsf{X}\varphi$, $\overleftarrow{\mathsf{X}}\varphi$, $[\varphi\ \underline{\mathsf{U}}\ \psi]$, $[\varphi\ \underline{\overleftarrow{\mathsf{U}}}\ \psi] \in \mathcal{L}_\omega$, if $\varphi, \psi \in \mathcal{L}_\omega$*
- *Given a finite set of variables Q with $Q \cap \mathcal{V} = \{\}$, a propositional formula $\Phi_\mathcal{I}$ over $Q \cup \mathcal{V}$, a propositional[2] formula $\Phi_\mathcal{R}$ over $Q \cup \mathcal{V} \cup \{\mathsf{X}v \mid v \in Q \cup \mathcal{V}\}$, and a $\mathcal{L}_\omega$ formula $\Phi_\mathcal{F}$ over $Q \cup \mathcal{V}$, then $\mathcal{A}_\exists\,(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi_\mathcal{F})$ is a $\mathcal{L}_\omega$ formula.*

The above logic is based on the usual temporal future time operators X and $[\cdot\ \underline{\mathsf{U}}\ \cdot]$, and their corresponding past time operators $\overleftarrow{\mathsf{X}}$ and $[\cdot\ \underline{\overleftarrow{\mathsf{U}}}\ \cdot]$. Moreover, it uses ω-automata as temporal operators to receive the full power of ω-regular languages. Note, that arbitrary $\mathcal{L}_\omega$ formulas are allowed as acceptance conditions in the automata expressions, which might be unusual, but simplifies the following explanations.

The semantics is given wrt. Kripke structures $\mathcal{K} = (\mathcal{I}, \mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$: $\mathcal{S}$ is a finite set of states, $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation, and $\mathcal{L} : \mathcal{S} \to 2^\mathcal{V}$ is a labeling function that maps a state $s \in \mathcal{S}$ to the set of variables $\mathcal{L}(s) \subseteq \mathcal{V}$ that hold on s. $\mathcal{I} \subseteq \mathcal{S}$ is the set of initial states, and $\mathcal{F} = \{F_1, \ldots, F_f\}$ is a finite set of sets of states F_i that are called fairness constraints. A path through $\mathcal{K}$ is a function $\pi : \mathbb{N} \to \mathcal{S}$ such that $\forall t.(\pi^{(t)}, \pi^{(t+1)}) \in \mathcal{R}$ holds. A path π is fair iff π visits every $F_i \in \mathcal{F}$ infinitely often.

$\mathcal{A}_\exists\,(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi_\mathcal{F})$ is an (existential) automaton formula that describes an ω-automaton in a symbolic manner: Q is the set of state variables, so the set of states corresponds with the powerset of Q. We identify any set $\vartheta \subseteq Q \cup \mathcal{V} \cup \{\mathsf{X}v \mid v \in Q \cup \mathcal{V}\}$ with a propositional interpretation that exactly assigns the variables of ϑ to true. With this view, the formula $\Phi_\mathcal{I}$ describes the set of the initial conditions: These are the sets $\vartheta \subseteq Q \cup \mathcal{V}$ that satisfy $\Phi_\mathcal{I}$. Similarly, $\Phi_\mathcal{R}$ describes the set of transitions. Intuitively, an existential automaton formula $\mathfrak{A}$ holds on a path π iff there is an accepting run for the trace $\mathcal{L}(\pi^{(0)})$, $\mathcal{L}(\pi^{(1)})$, …, through $\mathfrak{A}$. To define this formally, we

[2] For variables v, we often treat $\mathsf{X}v$ as a normal variable for propositional evaluations.

define the Kripke structure $\mathcal{K}_{\mathfrak{A}} = (\mathcal{I}_{\mathfrak{A}}, \mathcal{S}_{\mathfrak{A}}, \mathcal{R}_{\mathfrak{A}}, \mathcal{L}_{\mathfrak{A}}, \{\})$ for an automaton formula $\mathfrak{A} = \mathcal{A}_{\exists}(Q, \Phi_{\mathcal{I}}, \Phi_{\mathcal{R}}, \Phi_{\mathcal{F}})$ with $\mathcal{S}_{\mathfrak{A}} := 2^{Q \cup \mathcal{V}}$, $\mathcal{I}_{\mathfrak{A}} := \{\vartheta \subseteq Q \cup \mathcal{V} \mid \vartheta \models \Phi_{\mathcal{I}}\}$, $\mathcal{R}_{\mathfrak{A}} := \{(\vartheta_0, \vartheta_1) \subseteq (Q \cup \mathcal{V}) \times (Q \cup \mathcal{V}) \mid \vartheta_0 \cup \{\mathsf{X}v \mid v \in \vartheta_1\} \models \Phi_{\mathcal{R}}\}$, and $\mathcal{L}_{\mathfrak{A}}(s) := s$. Note that $\mathcal{K}_{\mathfrak{A}}$ is independent of the acceptance condition $\Phi_{\mathcal{F}}$; the effect of $\Phi_{\mathcal{F}}$ is considered in the definition of the semantics below:

Definition 2 (Semantics of $\mathcal{L}_\omega$). *Given a structure $\mathcal{K} = (\mathcal{I}, \mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$, a fair path π through $\mathcal{K}$, and a number $t \in \mathbb{N}$, the semantics of $\mathcal{L}_\omega$ formulas is recursively defined as follows:*

- $(\mathcal{K}, \pi, t) \models x$ *iff* $x \in \mathcal{L}(\pi^{(t)})$ *for each variable* x
- $(\mathcal{K}, \pi, t) \models \neg\varphi$ *iff not* $(\mathcal{K}, \pi, t) \models \varphi$
- $(\mathcal{K}, \pi, t) \models \varphi \vee \psi$ *iff* $(\mathcal{K}, \pi, t) \models \varphi$ *or* $(\mathcal{K}, \pi, t) \models \psi$
- $(\mathcal{K}, \pi, t) \models \mathsf{X}\varphi$ *iff* $(\mathcal{K}, \pi, t+1) \models \varphi$
- $(\mathcal{K}, \pi, t) \models \overleftarrow{\underline{\mathsf{X}}}\varphi$ *iff* $t = 0$ *or* $t > 0$ *and* $(\mathcal{K}, \pi, t-1) \models \varphi$
- $(\mathcal{K}, \pi, t) \models [\varphi \,\underline{\mathsf{U}}\, \psi]$ *iff there is a* $\delta \geq t$ *such that* $(\mathcal{K}, \pi, \delta) \models \psi$ *and for all* x *with* $t \leq x < \delta$, *we have* $(\mathcal{K}, \pi, x) \models \varphi$
- $(\mathcal{K}, \pi, t) \models [\varphi \,\overleftarrow{\underline{\mathsf{U}}}\, \psi]$ *iff there is a* $\delta \leq t$ *such that* $(\mathcal{K}, \pi, \delta) \models \psi$ *and for all* x *with* $\delta < x \leq t$, *we have* $(\mathcal{K}, \pi, x) \models \varphi$
- *Given* $\mathfrak{A} = \mathcal{A}_{\exists}(Q, \Phi_{\mathcal{I}}, \Phi_{\mathcal{R}}, \Phi_{\mathcal{F}})$ *with Kripke structure* $\mathcal{K}_{\mathfrak{A}} = (\mathcal{I}_{\mathfrak{A}}, \mathcal{S}_{\mathfrak{A}}, \mathcal{R}_{\mathfrak{A}}, \mathcal{L}_{\mathfrak{A}}, \{\})$, *we define* $(\mathcal{K}, \pi, t) \models \mathfrak{A}$ *iff there is fair path* ξ *through*[3] $\mathcal{K} \times \mathcal{K}_{\mathfrak{A}}$ *such that (1)* $\forall x.\mathsf{fst}(\xi^{(x)}) = \pi^{(x+t)}$, *(2)* $\mathsf{snd}(\xi^{(0)}) \in \mathcal{I}_{\mathfrak{A}}$, *and (3)* $(\mathcal{K} \times \mathcal{K}_{\mathfrak{A}}, \xi, 0) \models \Phi_{\mathcal{F}}$ *holds.*[4]

φ is generally valid if $(\mathcal{K}, \pi, t) \models \varphi$ holds on every path of every structure $\mathcal{K}$ for any position t. φ is initially valid if $(\mathcal{K}, \pi, 0) \models \varphi$ holds on every path of every structure $\mathcal{K}$.

Except for the automaton formulas, the above definition is done in the standard way and can be found in many related papers. Note that $\mathsf{snd}(\xi^{(0)})$ must be an initial state of $\mathcal{K}_{\mathfrak{A}}$, but $\mathsf{fst}(\xi^{(0)})$ is simply $\pi^{(t)}$ and not necessarily an initial state of $\mathcal{K}$.

Moreover, we use the following standard abbreviations: $\mathsf{G}a = \neg[1 \,\underline{\mathsf{U}}\, (\neg a)]$, $\mathsf{F}a = [1 \,\underline{\mathsf{U}}\, a]$, $[a \,\mathsf{U}\, b] = [a \,\underline{\mathsf{U}}\, b] \vee \mathsf{G}a$, $\overleftarrow{\underline{\mathsf{X}}}a = \neg\overleftarrow{\mathsf{X}}\neg a$, $\overleftarrow{\mathsf{G}}a = \neg[1 \,\overleftarrow{\underline{\mathsf{U}}}\, (\neg a)]$, $\overleftarrow{\mathsf{F}}a = [1 \,\overleftarrow{\underline{\mathsf{U}}}\, a]$, and $[a \,\overleftarrow{\mathsf{U}}\, b] = [a \,\overleftarrow{\underline{\mathsf{U}}}\, b] \vee \overleftarrow{\mathsf{G}}a$. We distinguish between strong and weak variants of temporal operators by underlining the strong variant (a strong variant requires that the event that is waited for actually will occur or has occurred, or in case of $\overleftarrow{\underline{\mathsf{X}}}$ that really a previous point of time exists). Moreover, we introduce further Boolean operators as $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi := \neg\varphi \vee \psi$, and $(\varphi = \psi) := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, with priorities $= \preceq \rightarrow \preceq \vee \preceq \wedge \preceq$ 'unary operators' ($\bigwedge_{i=0}^{n}$ and $\bigvee_{i=0}^{n}$ are also unary).

As an important notation, we write $\Phi\langle\varphi\rangle_x$ to denote that the variable x is replaced in Φ by the formula φ. This notation for substitution will be important in the following. Finally, we denote the length of a formula Φ, i.e., the number of its operators as $|\Phi|$.

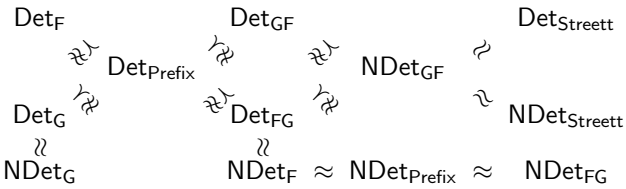
3 A Hierarchy of Temporal Logics

Several acceptance conditions are distinguished for defining different classes of ω-automata [36, 32]. Acceptance conditions of the form $\mathsf{G}\varphi_0$, $\mathsf{F}\varphi_0$, $\bigwedge_{i=0}^{n}(\mathsf{G}\varphi_i \vee \mathsf{F}\psi_i)$,

[3] Products of structures are defined in the usual way (see e.g., [27]).

[4] We assume the following definitions: $\mathsf{fst}((a,b)) = a$ and $\mathsf{snd}((a,b)) = b$.

$\mathsf{FG}\varphi_0$, $\mathsf{GF}\varphi_0$, and $\bigwedge_{i=0}^{n}(\mathsf{FG}\varphi_i \vee \mathsf{GF}\psi_i)$ determine the automaton classes $(\mathsf{N})\mathsf{Det}_{\mathsf{G}}$, $(\mathsf{N})\mathsf{Det}_{\mathsf{F}}$, $(\mathsf{N})\mathsf{Det}_{\mathsf{Prefix}}$, $(\mathsf{N})\mathsf{Det}_{\mathsf{FG}}$, $(\mathsf{N})\mathsf{Det}_{\mathsf{GF}}$, $(\mathsf{N})\mathsf{Det}_{\mathsf{Streett}}$, respectively, where all subformulas φ_i and ψ_i are propositional. Det_κ and NDet_κ denote thereby the sets of deterministic and nondeterministic ω-automata with acceptance condition of type κ. The expressiveness of these classes can be illustrated as follows, where $\mathcal{C}_1 \precapprox \mathcal{C}_2$ means that for any automaton in $\mathcal{C}_1$, there is an equivalent one in $\mathcal{C}_2$. Moreover, we define $\mathcal{C}_1 \approx \mathcal{C}_2 := \mathcal{C}_1 \precapprox \mathcal{C}_2 \wedge \mathcal{C}_2 \precapprox \mathcal{C}_1$ and $\mathcal{C}_1 \precnapprox \mathcal{C}_2 := \mathcal{C}_1 \precapprox \mathcal{C}_2 \wedge \neg(\mathcal{C}_1 \approx \mathcal{C}_2)$.



The above automaton hierarchy is closely related to the Borel hierarchy of topology [32]. As can be seen, it consists of six different classes, and each class has a deterministic representative. It can be shown that the nonemptiness problem for all automaton classes except for $(\mathsf{N})\mathsf{Det}_{\mathsf{GF}}$ and $(\mathsf{N})\mathsf{Det}_{\mathsf{Streett}}$ can be reduced to the alternation-free μ-calculus, whereas $(\mathsf{N})\mathsf{Det}_{\mathsf{GF}}$ and $(\mathsf{N})\mathsf{Det}_{\mathsf{Streett}}$ require μ-calculus formulas of alternation-depth 2. This is not only a theoretical issue, since the alternation-depth dramatically influences the runtime of the verification procedure (it reflects the number of nested fixpoint iterations of the model checking procedure).

Manna and Pnueli were the first who investigated a temporal logic hierarchy in analogy to the above automaton hierarchy [20]. However, they only considered very restricted normal forms, namely formulas that are obtained by replacing the subformulas φ_i and ψ_i in the acceptance conditions of the automata with formulas that contain only Boolean and past temporal operators. In the following definition, we present a completely new definition of a temporal logic hierarchy with syntactically much richer temporal logics. One can even show [28] that the future time fragments of these logics are as expressive as the logic themselves, so that past operators can be eliminated.

Definition 3 (Temporal Borel Classes). *We define the logics* TL_κ *for* $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{FG}, \mathsf{GF}, \mathsf{Streett}\}$ *by the following grammar rules, where* TL_κ *is the set of formulas that can be derived from the nonterminal* P_κ *($\mathcal{V}$ represents any variable $v \in \mathcal{V}$):*

$P_{\mathsf{G}} ::= \mathcal{V} \mid \neg P_{\mathsf{F}} \mid P_{\mathsf{G}} \wedge P_{\mathsf{G}} \mid P_{\mathsf{G}} \vee P_{\mathsf{G}}$ $\mid \overleftarrow{\mathsf{X}} P_{\mathsf{G}} \mid [P_{\mathsf{G}}\ \overleftarrow{\mathsf{U}}\ P_{\mathsf{G}}]$ $\mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{G}} \mid [P_{\mathsf{G}}\ \underline{\overleftarrow{\mathsf{U}}}\ P_{\mathsf{G}}]$ $\mid \mathsf{X} P_{\mathsf{G}} \mid [P_{\mathsf{G}}\ \mathsf{U}\ P_{\mathsf{G}}]$	$P_{\mathsf{F}} ::= \mathcal{V} \mid \neg P_{\mathsf{G}} \mid P_{\mathsf{F}} \wedge P_{\mathsf{F}} \mid P_{\mathsf{F}} \vee P_{\mathsf{F}}$ $\mid \overleftarrow{\mathsf{X}} P_{\mathsf{F}} \mid [P_{\mathsf{F}}\ \overleftarrow{\mathsf{U}}\ P_{\mathsf{F}}]$ $\mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{F}} \mid [P_{\mathsf{F}}\ \underline{\overleftarrow{\mathsf{U}}}\ P_{\mathsf{F}}]$ $\mid \mathsf{X} P_{\mathsf{F}} \mid [P_{\mathsf{F}}\ \underline{\mathsf{U}}\ P_{\mathsf{F}}]$
$P_{\mathsf{Prefix}} ::= P_{\mathsf{G}} \mid P_{\mathsf{F}} \mid \neg P_{\mathsf{Prefix}} \mid P_{\mathsf{Prefix}} \wedge P_{\mathsf{Prefix}} \mid P_{\mathsf{Prefix}} \vee P_{\mathsf{Prefix}}$	
$P_{\mathsf{GF}} ::= P_{\mathsf{Prefix}}$ $\mid \neg P_{\mathsf{FG}} \mid P_{\mathsf{GF}} \wedge P_{\mathsf{GF}} \mid P_{\mathsf{GF}} \vee P_{\mathsf{GF}}$ $\mid \overleftarrow{\mathsf{X}} P_{\mathsf{GF}} \mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{GF}} \mid \mathsf{X} P_{\mathsf{GF}}$ $\mid [P_{\mathsf{GF}}\ \overleftarrow{\mathsf{U}}\ P_{\mathsf{GF}}] \mid [P_{\mathsf{GF}}\ \underline{\overleftarrow{\mathsf{U}}}\ P_{\mathsf{GF}}]$ $\mid [P_{\mathsf{GF}}\ \mathsf{U}\ P_{\mathsf{GF}}] \mid [P_{\mathsf{GF}}\ \underline{\mathsf{U}}\ P_{\mathsf{F}}]$	$P_{\mathsf{FG}} ::= P_{\mathsf{Prefix}}$ $\mid \neg P_{\mathsf{GF}} \mid P_{\mathsf{FG}} \wedge P_{\mathsf{FG}} \mid P_{\mathsf{FG}} \vee P_{\mathsf{FG}}$ $\mid \overleftarrow{\mathsf{X}} P_{\mathsf{FG}} \mid \mathsf{X} P_{\mathsf{FG}} \mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{FG}}$ $\mid [P_{\mathsf{FG}}\ \overleftarrow{\mathsf{U}}\ P_{\mathsf{FG}}] \mid [P_{\mathsf{FG}}\ \underline{\overleftarrow{\mathsf{U}}}\ P_{\mathsf{FG}}]$ $\mid [P_{\mathsf{FG}}\ \underline{\mathsf{U}}\ P_{\mathsf{FG}}] \mid [P_{\mathsf{G}}\ \mathsf{U}\ P_{\mathsf{FG}}]$
$P_{\mathsf{Streett}} ::= P_{\mathsf{GF}} \mid P_{\mathsf{FG}} \mid \neg P_{\mathsf{Streett}} \mid P_{\mathsf{Streett}} \wedge P_{\mathsf{Streett}} \mid P_{\mathsf{Streett}} \vee P_{\mathsf{Streett}}$	

TL_G is the set of formulas where each occurrence of a weak/strong temporal future operator is positive/negative, and similarly, each occurrence of a weak/strong temporal future operator in TL_F is negative/positive. Hence, both logics are dual to each other, which means that one contains the negations of the other one. $\mathsf{TL}_{\mathsf{Prefix}}$ is the Boolean closure of TL_G (and TL_F). The logics TL_GF and TL_FG are constructed in the same way as TL_G and TL_F; there are two differences: (1) these logics allow occurrences of $\mathsf{TL}_{\mathsf{Prefix}}$ where otherwise variables would have been required in TL_G and TL_F, and (2) there are additional 'asymmetric' grammar rules. It can be easily proved that TL_GF and TL_FG are also dual to each other, and their intersection strictly contains $\mathsf{TL}_{\mathsf{Prefix}}$. Finally, $\mathsf{TL}_{\mathsf{Streett}}$ is the Boolean closure of TL_GF (and TL_FG).

Note that the 'asymmetric' grammar rules of the logics TL_GF and TL_FG can be eliminated due to the following equivalences: $[\varphi \,\underline{\mathsf{U}}\, \psi] = [\varphi \,\mathsf{U}\, \psi] \wedge \mathsf{F}\psi$ and $[\varphi \,\mathsf{U}\, \psi] = [\varphi \,\underline{\mathsf{U}}\, \psi] \vee \mathsf{G}\varphi$. In the following, we therefore neglect these asymmetric grammar rules to simplify the remaining considerations[5].

For the completeness of the logics TL_κ, we note that the logics TL_κ (syntactically) strictly contain the logics defined in [20] that are known to be complete with respect to counter-free Det_κ automata. Hence, it is possible to translate any counter-free Det_κ automaton to an equivalent TL_κ formula, the other direction is proved in the following.

4 The Basic Translation to ω-Automata

Translations from temporal logics to equivalent ω-automata have been intensively studied. Common to most procedures is the consideration of the truth values of all subformulas of the given formula. In particular, one considers the set of *elementary subformulas* of a given formula Φ which is essentially the set of all subformulas of Φ that start with a temporal operator. The states of the ω-automaton that is to be constructed consist then of the truth values of these elementary formulas: If Φ has the elementary formulas $\{\varphi_1, \ldots, \varphi_n\}$, we need n state variables $\{q_1, \ldots, q_n\}$ to encode the state set, and therefore already see where the exponential blow-up of the automaton comes from. Clearly, for any run through the automaton, we want that $q_i = \varphi_i$ holds. For this reason, the transition relation of the automaton must respect the semantics of the temporal operators that occur in φ_i. The following theorem shows however that it is not enough to only follow the recursion laws of the operators:

Theorem 1. *Given a formula Φ with some occurrences of a variable x, and propositional formulas φ and ψ, the following equations are valid :*

$$\begin{aligned}
\mathcal{A}_\exists\left(\{q\}, 1, \mathsf{X}q = \varphi, \Phi\langle q\rangle_x\right) &= \Phi\langle \overleftarrow{\mathsf{X}}\varphi\rangle_x \vee \Phi\langle \overleftarrow{\underline{\mathsf{X}}}\varphi\rangle_x \\
\mathcal{A}_\exists\left(\{q\}, 1, \mathsf{X}q = \psi \vee \varphi \wedge q, \Phi\langle \psi \vee \varphi \wedge q\rangle_x\right) &= \Phi\langle[\varphi \,\overleftarrow{\mathsf{U}}\, \psi]\rangle_x \vee \Phi\langle[\varphi \,\overleftarrow{\underline{\mathsf{U}}}\, \psi]\rangle_x \\
\mathcal{A}_\exists\left(\{q\}, 1, q = \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x\right) &= \Phi\langle[\varphi \,\mathsf{U}\, \psi]\rangle_x \vee \Phi\langle[\varphi \,\underline{\mathsf{U}}\, \psi]\rangle_x
\end{aligned}$$

As strong and weak operators fulfill exactly the same recursion laws, they can not be distinguished by transition relations alone. The next theorem shows that we can select the strong or the weak variant by either adding suitable initialization conditions or fairness constraints (for past and future operators, respectively).

[5] A simple rewriting procedure would however blow-up the formula which is not necessary when the translation procedures directly support the full logics [28].

Theorem 2 (Translating Temporal Logic to ω-Automata). *Given a formula Φ, a variable x, and propositional formulas φ and ψ, the following equations are valid:*

$$\begin{aligned}
\Phi\langle \overleftarrow{\mathsf{X}}\varphi\rangle_x &= \mathcal{A}_\exists\,(\{q\}, q, \mathsf{X}q = \varphi, \Phi\langle q\rangle_x)\\
\Phi\langle \overleftarrow{\underline{\mathsf{X}}}\varphi\rangle_x &= \mathcal{A}_\exists\,(\{q\}, \neg q, \mathsf{X}q = \varphi, \Phi\langle q\rangle_x)\\
\Phi\langle [\varphi\ \overleftarrow{\mathsf{U}}\ \psi]\rangle_x &= \mathcal{A}_\exists\,(\{q\}, q, \mathsf{X}q = \psi \vee \varphi \wedge q, \Phi\langle \psi \vee \varphi \wedge q\rangle_x)\\
\Phi\langle [\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi]\rangle_x &= \mathcal{A}_\exists\,(\{q\}, \neg q, \mathsf{X}q = \psi \vee \varphi \wedge q, \Phi\langle \psi \vee \varphi \wedge q\rangle_x)\\
\Phi\langle \mathsf{X}\varphi\rangle_x &= \mathcal{A}_\exists\,(\{q\}, 1, q = \mathsf{X}\varphi, \Phi\langle q\rangle_x)\\
\Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x &= \mathcal{A}_\exists\,(\{q\}, 1, q = \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x \wedge \mathsf{GF}[\varphi \rightarrow q])\\
\Phi\langle [\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x &= \mathcal{A}_\exists\,(\{q\}, 1, q = \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x \wedge \mathsf{GF}[q \rightarrow \psi])
\end{aligned}$$

There is a subtlety concerning the X operator: If φ contains occurrences of input variables, the transition relation of the automaton refers to the next input, which is normally not allowed for automata[6]. Theorem 2 can already be used to translate any temporal logic formula Φ to an equivalent ω-automaton $\mathfrak{A}_\Phi$, if the laws are applied in a bottom-up traversal over the syntax tree of Φ to abbreviate any elementary formula by a propositional one [29].

Theorem 3 (Basic Translation). *Given any formula $\Phi \in \mathsf{LTL}$, the laws given in theorem 2 can be used to compute an equivalent* $\mathsf{NDet}_{\mathsf{Streett}}$ *automaton $\mathfrak{A}$ in time $O(|\Phi|)$.*

A machine-checked proof of the above theorem can be found in [29]; the sources are available in the newest HOL98 distributions. Although the automaton may have in the worst case $O(2^{|\Phi|})$ states, we still compute it in linear time, since we use a *symbolic representation*, that is directly usable for symbolic model checking.

5 Improving the Basic Translation

5.1 Exploiting the Monotonicity of Temporal Operators

Reconsider the translation procedure of the previous section: If we omit the fairness constraints, then we leave it unspecified whether the operator that is on top of the abbreviated formula was a weak or a strong one: For example, the automaton formula $\mathcal{A}_\exists\,(\{q\}, 1, q = \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x)$ is equivalent to $\Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x \vee \Phi\langle [\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x$ (cf. theorem 1). Our aim is therefore to find a simple condition so that $\Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x \vee \Phi\langle [\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x = \Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x$ holds. As $p \vee q = q$ is equivalent to $p \rightarrow q$, it follows that we must find a criterion for Φ so that $\Phi\langle [\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x \rightarrow \Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x$ holds. As $[\varphi\ \underline{\mathsf{U}}\ \psi] \rightarrow [\varphi\ \mathsf{U}\ \psi]$ holds, this means in turn that we must study the monotonicity of temporal/Boolean operators (we define the partial order $\varphi \preceq \psi$ as $\models \mathsf{G}[\varphi \rightarrow \psi]$).

It is not difficult to prove that all of our temporal operators are monotonic in all arguments. For example, $\mathsf{G}[\underline{\alpha} \rightarrow \alpha]$, and $\mathsf{G}[\underline{\beta} \rightarrow \beta]$ implies $\mathsf{G}\left(\left[\underline{\alpha}\ \underline{\mathsf{U}}\ \underline{\beta}\right] \rightarrow [\alpha\ \underline{\mathsf{U}}\ \beta]\right)$. Hence, checking the monotonicity of a temporal formula $\Phi\langle\varphi\rangle_x$ in the argument x reduces to checking whether φ occurs only under an even or odd number of negations. To be precise and concise, we use the notion of positive/negative occurrences in a temporal logic formula: An occurrence is positive/negative iff it appears under an even/odd number of negation symbols (we exclude implications and equivalences at the moment). It is straightforward to prove the following theorem:

[6] The alternative $\Phi\langle \mathsf{X}\varphi\rangle_x = \mathcal{A}_\exists\,(\{q_0, q_1\}, 1, (q_0 = \varphi) \wedge (q_1 = \mathsf{X}q_0), \Phi\langle q_1\rangle_x)$ circumvents this.

function $\mathsf{TopProp}_\sigma(\Phi)$
  case Φ **of**
    $\mathsf{is\_prop}(\Phi)$: **return** $\mathcal{A}_\exists\,(\{\},1,1,\{\},\Phi)$;
    $\neg\varphi$: $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_{\neg\sigma}(\varphi)$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\neg\varphi')$;
    $\varphi \wedge \psi$: $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi')$;
    $\varphi \vee \psi$: $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \vee \psi')$;
    $\mathsf{X}\varphi$: $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_\sigma(\varphi)$; $q = \mathsf{new\_var}$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi \cup \{q\},\mathcal{I}_\varphi,\mathcal{R}_\varphi \wedge (q = \mathsf{X}\varphi')\,,\mathcal{F}_\varphi,q)$;
    $[\varphi\ \mathsf{U}\ \psi]$: $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
      $q = \mathsf{new\_var}$; $\mathcal{R}_q = [q = \psi' \vee \varphi' \wedge \mathsf{X}q]$;
      $\mathcal{F}_q$ = **if** σ **then** $\{\}$ **else** $\{\mathsf{GF}[\varphi' \rightarrow q]\}$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi \cup \mathcal{F}_q,q)$;
    $[\varphi\ \underline{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
      $q = \mathsf{new\_var}$; $\mathcal{R}_q = [q = \psi' \vee \varphi' \wedge \mathsf{X}q]$;
      $\mathcal{F}_q$ = **if** σ **then** $\{\mathsf{GF}[q \rightarrow \psi']\}$ **else** $\{\}$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi \cup \mathcal{F}_q,q)$;
    $\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_\sigma(\varphi)$; $q = \mathsf{new\_var}$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi \cup \{q\},\mathcal{I}_\varphi \wedge q,\mathcal{R}_\varphi \wedge (\mathsf{X}q = \varphi')\,,\mathcal{F}_\varphi,q)$;
    $\underline{\overleftarrow{\mathsf{X}}}\varphi$: $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_\sigma(\varphi)$; $q = \mathsf{new\_var}$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\varphi \cup \{q\},\mathcal{I}_\varphi \wedge \neg q,\mathcal{R}_\varphi \wedge (\mathsf{X}q = \varphi')\,,\mathcal{F}_\varphi,q)$;
    $[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
      $q = \mathsf{new\_var}$; $r_q = \psi' \vee \varphi' \wedge q$; $\mathcal{R}_q = [\mathsf{X}q = r_q]$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi \wedge q,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi,r_q)$;
    $[\varphi\ \underline{\overleftarrow{\mathsf{U}}}\ \psi]$: $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
      $q = \mathsf{new\_var}$; $r_q = \psi' \vee \varphi' \wedge q$; $\mathcal{R}_q = [\mathsf{X}q = r_q]$;
      return $\mathcal{A}_\exists\,(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi \wedge \neg q,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi,r_q)$;
  end case
end function

Fig. 1. Improving the Translation from Temporal Logic to ω-Automata by Considering the Monotonicity of Operators

Theorem 4 (Translation to ω-Automata wrt. Positive/Negative Occurrences). *Given a formula Φ, a variable x, and propositional formulas φ and ψ. Then the following equation is generally valid, provided that any occurrence of x in Φ is positive:*

$$\Phi\langle[\varphi\ \mathsf{U}\ \psi]\rangle_x = \mathcal{A}_\exists\,(\{q\},1,q = \psi \vee \varphi \wedge \mathsf{X}q,\Phi\langle q\rangle_x)$$

If, on the other hand, any occurrence of x in Φ is negative, then the following holds:

$$\Phi\langle[\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x = \mathcal{A}_\exists\,(\{q\},1,q = \psi \vee \varphi \wedge \mathsf{X}q,\Phi\langle q\rangle_x)$$

Hence, fairness constraints need not be generated for positive/negative occurrences of weak/strong temporal future operators. This improves the translations of [3, 6, 18, 29], however, [8] already uses this improvement. An algorithm is given in Figure 1, where σ

denotes the signum of the occurrence. For convenience, we extended automaton formulas in the algorithm by an additional argument for the constraints: $\mathcal{A}_\exists(\mathcal{Q},\mathcal{I},\mathcal{R},\mathcal{F},\Phi) = \mathcal{A}_\exists\left(\mathcal{Q},\mathcal{I},\mathcal{R},\Phi\wedge\bigwedge_{\xi\in\mathcal{F}}\xi\right)$, and $\mathcal{A}_\exists(Q_1,\mathcal{I}_1,\mathcal{R}_1,\mathcal{F}_1,\Phi_1)\times\mathcal{A}_\exists(Q_2,\mathcal{I}_2,\mathcal{R}_2,\mathcal{F}_2,\Phi_2)$ is defined as $\mathcal{A}_\exists(Q_1\cup Q_2,\mathcal{I}_1\wedge\mathcal{I}_2,\mathcal{R}_1\wedge\mathcal{R}_2,\mathcal{F}_1\cup\mathcal{F}_2,\Phi_1\wedge\Phi_2)$.

Theorem 5 (Correctness of $\mathsf{TopProp}_\sigma(\Phi)$). *For any $\Phi\in\mathsf{LTL}$ and the automaton formula $\mathcal{A}_\exists(\mathcal{Q},\mathcal{I},\mathcal{R},\mathcal{F},\Phi_0) = \mathsf{TopProp}_\sigma(\Phi)$ obtained by the algorithm given in Figure 1, we have the following facts:*

(1) $\mathsf{TopProp}_\sigma(\Phi)$ *runs in time* $O(|\Phi|)$.
(2) Φ_0 *is propositional.*
(3) *Each* ξ_i *is of the form* $\mathsf{GF}\xi_i'$ *where* ξ_i' *is propositional.*
(4) *For* $\sigma=1$, *the equation* $\Phi = \mathcal{A}_\exists(\mathcal{Q},\mathcal{I},\mathcal{R},\mathcal{F},\Phi_0)$ *is initially valid.*
(5) *For* $\sigma=0$, *the equation* $\neg\Phi = \mathcal{A}_\exists(\mathcal{Q},\mathcal{I},\mathcal{R},\mathcal{F},\neg\Phi_0)$ *is initially valid.*

5.2 Considering Finite Intervals of Interest

We have seen above that the basic translation as given in Section 4 can be improved by exploiting the monotonicity of operators. This allows one to avoid the introduction of fairness constraints for all positive/negative occurrences of weak/strong operators. As a second improvement, we will now show that in the remaining situations, the operator strength can often be fixed by a simpler *reachability constraint* of the form $\mathsf{F}\varphi$ instead of a more expensive fairness constraint $\mathsf{GF}\varphi$.

The key is thereby the following: Assume Φ contains only positive/negative occurrences of strong/weak temporal future operators. For any path π with $(\mathcal{K},\pi,0)\models\Phi$, we must evaluate the subformulas of Φ *for only finitely many points of time*, i.e., all models of Φ do only depend on a finite prefix. Hence, it is not necessary to define state variables q_φ and q_ψ such that the equations $q_\varphi=\varphi$ and $q_\psi=\psi$ hold *for all points of time*. Instead, it is completely sufficient that these equations hold long enough, i.e., for some finite interval. The key to our next improvement is therefore the following lemma:

Lemma 1 (Solutions of $q=\psi\vee\varphi\wedge\mathsf{X}q$). *Given that the formula $\mathsf{G}[q=\psi\vee\varphi\wedge\mathsf{X}q]$ initially holds, then the following equations initially holds:*

$$\mathsf{G}\left[(\mathsf{F}(q\rightarrow\psi))\rightarrow\left(\overleftarrow{\mathsf{G}}(q=[\varphi\,\underline{\mathsf{U}}\,\psi])\right)\right]\qquad \mathsf{G}\left[(\neg\mathsf{F}(q\rightarrow\psi))\rightarrow(\mathsf{G}(q=[\varphi\,\mathsf{U}\,\psi]))\right]$$

This lemma is used as follows: Whenever a state variable q always satisfies the fixpoint equation $q=\psi\vee\varphi\wedge\mathsf{X}q$, and at some point of time $\mathsf{F}(q\rightarrow\psi)$ holds, then it follows that q is uniquely defined up to this point of time, since $q=[\varphi\,\underline{\mathsf{U}}\,\psi]$ holds up to this point of time. Hence, we can still abbreviate an elementary subformula $[\varphi\,\underline{\mathsf{U}}\,\psi]$ with a new state variable q by adding the equation $q=\psi\vee\varphi\wedge\mathsf{X}q$ to the transition relation. At the end of the interval I where we need to evaluate $[\varphi\,\underline{\mathsf{U}}\,\psi]$, we demand that $\mathsf{F}(q\rightarrow\psi)$ must hold (this can always be demanded, since $\mathsf{F}([\varphi\,\underline{\mathsf{U}}\,\psi]\rightarrow\psi)$ is generally valid).

It remains to define for each subformula the maximal point of time, where it must be evaluated. It is to be noted that nestings of temporal operators extend these intervals, since at the end of the interval of the outermost operator, we need to evaluate the inner ones. An algorithm that keeps track of this, is given in Figure 2.

function $\mathsf{BorelFG}_\sigma(\Phi)$
case Φ **of**
$\mathsf{is\_prop}(\Phi)$: **return** $\mathcal{A}_\exists(\{\}, 1, 1, \{\}, \Phi)$;
$\neg\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{BorelFG}_{\neg\sigma}(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \neg\varphi')$;
$\varphi \wedge \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{BorelFG}_\sigma(\varphi) \times \mathsf{BorelFG}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi')$;
$\varphi \vee \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{BorelFG}_\sigma(\varphi) \times \mathsf{BorelFG}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \vee \psi')$;
$\mathsf{X}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{BorelFG}_\sigma(\varphi)$; $q = \mathsf{new\_var}$;
if $\mathcal{F}_\varphi \neq \{\}$ **then** $\mathcal{F}_\varphi := \{\mathsf{X}\bigwedge_{\xi\in\mathcal{F}_\Phi}\xi\}$ **end**;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi, \mathcal{R}_\varphi \wedge (q = \mathsf{X}\varphi'), \mathcal{F}_\varphi, q)$;
$[\varphi\ \mathsf{U}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{BorelFG}_\sigma(\varphi) \times \mathsf{BorelFG}_\sigma(\psi)$;
$q = \mathsf{new\_var}$; $\mathcal{R}_q = [q = \psi' \vee \varphi' \wedge \mathsf{X}q]$;
$\mathcal{F}_q$ = **if** σ **then** $\{\}$ **else** $\left\{\mathsf{F}\left[(\varphi' \to q) \wedge \bigwedge_{\xi\in\mathcal{F}_\Phi}\xi\right]\right\}$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_q, q)$;
$[\varphi\ \underline{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{BorelFG}_\sigma(\varphi) \times \mathsf{BorelFG}_\sigma(\psi)$;
$q = \mathsf{new\_var}$; $\mathcal{R}_q = [q = \psi' \vee \varphi' \wedge \mathsf{X}q]$;
$\mathcal{F}_q$ = **if** σ **then** $\left\{\mathsf{F}\left[(q \to \psi') \wedge \bigwedge_{\xi\in\mathcal{F}_\Phi}\xi\right]\right\}$ **else** $\{\}$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_q, q)$;
$\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{BorelFG}_\sigma(\varphi)$; $q = \mathsf{new\_var}$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi \wedge q, \mathcal{R}_\varphi \wedge (\mathsf{X}q = \varphi'), \mathcal{F}_\varphi, q)$;
$\underline{\overleftarrow{\mathsf{X}}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{BorelFG}_\sigma(\varphi)$; $q = \mathsf{new\_var}$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi \wedge \neg q, \mathcal{R}_\varphi \wedge (\mathsf{X}q = \varphi'), \mathcal{F}_\varphi, q)$;
$[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{BorelFG}_\sigma(\varphi) \times \mathsf{BorelFG}_\sigma(\psi)$;
$q = \mathsf{new\_var}$; $r_q = \psi' \vee \varphi' \wedge q$; $\mathcal{R}_q = [\mathsf{X}q = r_q]$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi \wedge q, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, r_q)$;
$[\varphi\ \underline{\overleftarrow{\mathsf{U}}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{BorelFG}_\sigma(\varphi) \times \mathsf{BorelFG}_\sigma(\psi)$;
$q = \mathsf{new\_var}$; $r_q = \psi' \vee \varphi' \wedge q$; $\mathcal{R}_q = [\mathsf{X}q = r_q]$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi \wedge \neg q, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, r_q)$;
end case
end function

Fig. 2. Considering Finite Intervals of Interest

Theorem 6 (Correctness of $\mathsf{BorelFG}_\sigma(\Phi)$). *For any $\Phi \in \mathsf{TL}_{\mathsf{FG}}$ and the automaton formula $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0) = \mathsf{BorelFG}_\sigma(\Phi)$ obtained by the algorithm of Figure 2, the following holds:*

(1) $\mathsf{BorelFG}_\sigma(\Phi)$ *runs in time $O(|\Phi|)$.*
(2) *Φ_0 is propositional.*
(3) *Each $\xi \in \mathcal{F}$ can be derived from the nonterminal C_F by the following grammar:*

$$P_{\mathsf{Prop}} ::= \mathcal{V} \mid \neg P_{\mathsf{Prop}} \mid P_{\mathsf{Prop}} \wedge P_{\mathsf{Prop}} \mid P_{\mathsf{Prop}} \vee P_{\mathsf{Prop}} \mid P_{\mathsf{Prop}} \to P_{\mathsf{Prop}}$$
$$C_\mathsf{F} ::= \mathsf{F}P_{\mathsf{Prop}} \mid \mathsf{F}(P_{\mathsf{Prop}} \wedge C_\mathsf{F}) \mid \mathsf{X}C_\mathsf{F} \mid C_\mathsf{F} \wedge C_\mathsf{F}$$

(4) *For $\sigma = 1$, the equation $\Phi = \mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0)$ is initially valid.*
(5) *For $\sigma = 0$, the equation $\neg\Phi = \mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \neg\Phi_0)$ is initially valid.*

For example, the formula $[[\varphi \,\underline{\mathsf{U}}\, \psi] \,\underline{\mathsf{U}}\, \gamma]$ is translated by introducing new state variables p,q with appropriate transition relations and the reachability constraint $\mathcal{F} = \{\mathsf{F}\,[(q \rightarrow \gamma) \wedge \mathsf{F}(p \rightarrow \psi)]\}$. For the proof of the above theorem, we used initially valid formulas of the following form to establish the invariants:

$$\left(\begin{array}{l} \mathsf{G}(\varphi \rightarrow \varphi_0) \wedge \mathsf{G}\left(\zeta_\varphi \rightarrow \overleftarrow{\mathsf{G}}\,[\varphi_0 = \varphi]\right) \wedge \\ \mathsf{G}(\psi \rightarrow \psi_0) \wedge \mathsf{G}\left(\zeta_\psi \rightarrow \overleftarrow{\mathsf{G}}\,[\psi_0 = \psi]\right) \wedge \\ \mathsf{G}\,[q = \psi_0 \vee \varphi_0 \wedge \mathsf{X}q] \rightarrow \mathsf{G}\left(\mathsf{F}\,([q \rightarrow \psi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \rightarrow \overleftarrow{\mathsf{G}}\,(q = [\varphi \,\underline{\mathsf{U}}\, \psi])\right) \end{array}\right)$$

The above formula is used in the induction step for $[\varphi \,\underline{\mathsf{U}}\, \psi]$. ζ_φ and ζ_ψ are thereby the reachability constraints that have been obtained from the translation of the subformulas φ and ψ. Due to the definition of $\mathsf{TL}_{\mathsf{FG}}$, we always have the monotonicity conditions $\mathsf{G}(\varphi \rightarrow \varphi_0)$ and $\mathsf{G}(\psi \rightarrow \psi_0)$, so that the conjunction of the transition relation with $q = \psi_0 \vee \varphi_0 \wedge \mathsf{X}q$ will fix q as desired whenever $\mathsf{F}\,([q \rightarrow \psi_0] \wedge \zeta_\varphi \wedge \zeta_\psi)$ holds.

5.3 Translation by Closures

The automaton obtained by the function BorelFG does not right fall into one of the automaton classes. However, it is easily seen that we can reduce it to either a $\mathsf{NDet}_{\mathsf{F}}$ or an $\mathsf{NDet}_{\mathsf{FG}}$ automaton. For reasons of efficiency[7], we choose the latter class for a further translation. This further translation is based on a simple bottom-up rewriting with the equations of the following lemma:

Lemma 2 (Closures of $\mathsf{NDet}_{\mathsf{FG}}$). *Given a propositional formula φ, and $\mathsf{NDet}_{\mathsf{FG}}$ automata $\mathfrak{A}_\Phi = \mathcal{A}_\exists\,(Q_\Phi, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \mathsf{FG}\Phi_\mathcal{F})$ and $\mathfrak{A}_\Psi = \mathcal{A}_\exists\,(Q_\Psi, \Psi_\mathcal{I}, \Psi_\mathcal{R}, \mathsf{FG}\Psi_\mathcal{F})$ with $Q_\Phi \cap Q_\Psi = \{\}$, the following equations are generally valid:*

- $\mathsf{F}\varphi = \mathcal{A}_\exists\,(\{q\}, \neg q, [\neg q \wedge \neg\varphi \wedge \neg\mathsf{X}q] \vee [\neg q \wedge \varphi \wedge \mathsf{X}q] \vee [q \wedge \mathsf{X}q]\,, \mathsf{FG}q)$
- $\mathsf{F}\,(\varphi \wedge \mathfrak{A}_\Phi) = \mathcal{A}_\exists \left(\begin{array}{l} Q \cup \{p\}, \neg p \vee \Phi_\mathcal{I}, \\ [\neg p \wedge \neg\mathsf{X}p] \vee \\ [\neg p \wedge \mathsf{X}\,(p \wedge \Phi_\mathcal{I} \wedge \varphi)] \vee \\ [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p]\,, \\ \mathsf{FG}(p \wedge \Phi_\mathcal{F}) \end{array}\right)$
- $\mathsf{X}\,[\mathfrak{A}_\Phi] = \mathcal{A}_\exists \left(\begin{array}{l} Q \cup \{p\}, \neg p, \\ [\neg p \wedge \mathsf{X}p \wedge \mathsf{X}\Phi_\mathcal{I}] \vee [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p]\,, \\ \mathsf{FG}\Phi_\mathcal{F} \end{array}\right)$
- $\varphi \wedge \mathfrak{A}_\Phi = \mathcal{A}_\exists\,(Q_\Phi, \Phi_\mathcal{I} \wedge \varphi, \Phi_\mathcal{R}, \mathsf{FG}\Phi_\mathcal{F})$
- $\mathfrak{A}_\Phi \wedge \mathfrak{A}_\Psi = \mathcal{A}_\exists\,(Q_\Phi \cup Q_\Psi, \Phi_\mathcal{I} \wedge \Psi_\mathcal{I}, \Phi_\mathcal{R} \wedge \Psi_\mathcal{R}, \mathsf{FG}(\Phi_\mathcal{F} \wedge \Psi_\mathcal{F}))$

The above equations are not hard to prove. The reader is asked to draw the transition systems to convince himself/herself of the correctness. *Using* BorelFG *and the above closures, we can now translate any* $\mathsf{TL}_{\mathsf{FG}}$ *formula to an equivalent* $\mathsf{NDet}_{\mathsf{FG}}$ *automaton with linear runtime and memory requirements (cf. Figure 3).*

[7] We only need the conjunction of automata, and never the disjunction. As $\mathsf{FG}\varphi \wedge \mathsf{FG}\psi = \mathsf{FG}\,(\varphi \wedge \psi)$ is valid, we can easily compute the conjunction of $\mathsf{NDet}_{\mathsf{FG}}$ automata, which is not so simple for $\mathsf{NDet}_{\mathsf{F}}$ automata.

6 The Final Translation

The algorithms of the previous sections already allow us to efficiently translate any $\mathsf{TL}_{\mathsf{FG}}$ formula to an equivalent $\mathsf{NDet}_{\mathsf{FG}}$ automaton with only linear runtime and memory requirements. This is important since each formula that can be translated to the alternation-free μ-calculus has an equivalent $\mathsf{TL}_{\mathsf{FG}}$ formula, since it can be translated to $\mathsf{NDet}_{\mathsf{FG}}$. Therefore, with symbolic model checking, these formulas can be more efficiently checked than arbitrary LTL formulas, and our experiments show that the verification procedures are almost comparable to CTL model checking (see also [33]).

The translation procedure for $\mathsf{TL}_{\mathsf{FG}}$ can also be used to enhance the translation of arbitrary LTL formulas. To see this, it is convenient to use the notion of *templates*: Φ is a template of Ψ, if Ψ can be obtained from Φ by replacing some occurrences of variables in Φ with some other formulas (Φ matches with Ψ). It is straightforward to compute from any temporal logic formula Φ the largest template that belongs to $\mathsf{TL}_{\mathsf{FG}}$. To perform this computation, we simply traverse the syntax tree of Φ and abbreviate any subformula that violates the grammar rules of $\mathsf{TL}_{\mathsf{FG}}$ by a new state variable by application of TopProp to that subformula (having this view, TopProp is also an extraction procedure that extracts the largest propositional template). The resulting function TopFG is very similar to TopProp, and also runs in linear time [28].

Theorem 7 (Algorithm TopFG). *There is an algorithm* TopFG *such that for any temporal logic formula $\Phi \in \mathsf{LTL}$, and the automaton formula $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0) = \mathsf{TopFG}_\sigma(\Phi)$, the following holds:*

(1) $\mathsf{TopFG}_\sigma(\Phi)$ *runs in time $O(|\Phi|)$.*
(2) *If $\sigma = 1$ holds, we have $\Phi_0 \in \mathsf{TL}_{\mathsf{FG}}$, otherwise, we have $\Phi_0 \in \mathsf{TL}_{\mathsf{GF}}$.*
(3) *Each $\xi \in \mathcal{F}$ is of the form $\mathsf{GF}\xi'$ where ξ' is propositional.*
(4) *For $\sigma = 1$, the equation $\Phi = \mathcal{A}_\exists\left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \Phi_0 \wedge \bigwedge_{\xi\in\mathcal{F}} \xi\right)$ is initially valid.*
(5) *For $\sigma = 0$, the equation $\neg\Phi = \mathcal{A}_\exists\left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \neg\Phi_0 \wedge \bigwedge_{\xi\in\mathcal{F}} \xi\right)$ is initially valid.*

Given any formula $\Phi \in \mathsf{LTL}$, we can therefore apply the function TopFG to extract the largest template of Φ that belongs to $\mathsf{TL}_{\mathsf{FG}}$. This template is then translated by BorelFG and the closure equations to an equivalent $\mathsf{NDet}_{\mathsf{FG}}$ automaton (without fairness constraints). Note that TopFG still introduces fairness constraints, but BorelFG does not.

This translation procedure can be further improved. To explain this, we need to note that the final automaton problems can be reduced to fair CTL model checking problems due to the following lemma, where E is the existential path quantifier ($\mathsf{E}\varphi$ holds on a state s if there is a fair path starting in s that satisfies φ):

Lemma 3 (Reducing ω-Automata Model Checking to CTL Model Checking). *Given an automaton formula $\mathfrak{A} = \mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F})$ so that $\mathcal{F}$ does not contain past temporal operators. Then, the following is equivalent for any structure $\mathcal{K}$, and any state s of $\mathcal{K}$:*

- $(\mathcal{K}, s) \models \mathsf{E}\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F})$
- *there is a $\vartheta \subseteq \mathcal{Q} \cup \mathcal{V}$ such that $(\mathcal{K} \times \mathcal{K}_\mathfrak{A}, (s, \vartheta)) \models \mathcal{I} \wedge \mathsf{E}\mathcal{F}$ and $\mathcal{L}(s) = \vartheta \cap \mathcal{V}$*

function Automaton$(OnlyMono, useBorel, toCTL, \Phi)$
  if *OnlyMono* **then return** $\mathsf{TopProp}_1(\Phi)$ **end**;
  $\mathcal{A}_\exists\,(\mathcal{Q}_0, \mathcal{I}_0, \mathcal{R}_0, \mathcal{F}_0, \Phi_0) \equiv \mathsf{TopFG}_1(\Phi)$;
  if *useBorel* **then** // *translate to reachability automaton*
    $\mathcal{A}_\exists\,(\mathcal{Q}_1, \mathcal{I}_1, \mathcal{R}_1, \mathcal{F}_1, \Phi_1) \equiv \mathsf{BorelFG}_1(\Phi_0)$;
    $\mathcal{I}_1 := \mathcal{I}_1 \wedge \Phi_1$;
    $\Phi_1 := \bigwedge_{\xi \in \mathcal{F}_1} \xi$;
  else // *extract top-level* $\mathsf{TL_{PE}}$ *formula*
    $\mathcal{A}_\exists\,(\mathcal{Q}_1, \mathcal{I}_1, \mathcal{R}_1, \Phi_1) \equiv \mathsf{TopPE}_1(\Phi_0)$;
  end;
  // *in any case, we now have* $\Phi_1 \in \mathsf{TL_{PE}}$
  if *toCTL* **then** // *translate to* CTL
    $\Phi_2 := \mathsf{LeftCTL2CTL}(\Phi_1)$;
    return $\mathcal{A}_\exists\,(\mathcal{Q}_0 \cup \mathcal{Q}_1, \mathcal{I}_0 \wedge \mathcal{I}_1, \mathcal{R}_0 \wedge \mathcal{R}_1, \mathcal{F}_0, \Phi_2)$;
  else // *apply closure theorems*
    $\mathcal{A}_\exists\,(\mathcal{Q}_2, \mathcal{I}_2, \mathcal{R}_2, \Phi_{\mathsf{FG}}) \equiv \mathsf{close}(\Phi_1)$;
    return $\mathcal{A}_\exists\,(\mathcal{Q}_0 \cup \mathcal{Q}_1 \cup \mathcal{Q}_2, \mathcal{I}_0 \wedge \mathcal{I}_1 \wedge \mathcal{I}_2, \mathcal{R}_0 \wedge \mathcal{R}_1 \wedge \mathcal{R}_2, \mathcal{F}_0, \Phi_{\mathsf{FG}})$;
end function

Fig. 3. The Final Translation from LTL to ω-Automata or CTL Model Checking

Hence a reduction to the standard automaton classes is not necessary. *Instead, a reduction to an 'ω-automaton' $\mathcal{A}_\exists\,(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F})$ would be sufficient so that* $\mathsf{E}\mathcal{F}$ *can be easily translated to a* CTL *formula*. For our model checking tool, this view offers another alternative to the translation to simple ω-automata that is already contained in the algorithm of Figure 3: Using $OnlyMono = 1$ will simply call TopProp for the translation. Otherwise, we extract the largest template Φ_0 of Φ that belongs to $\mathsf{TL_{FG}}$. Using the flag *useBorel*, we then have the choice to either apply BorelFG to compute an automaton $\mathcal{A}_\exists\,(\mathcal{Q}_1, \mathcal{I}_1, \mathcal{R}_1, \mathcal{F}_1, \Phi_1)$ where Φ_1 is propositional and the constraints of $\mathcal{F}_1$ obey the grammar rules of theorem 6, or to reduce Φ_1 to an automaton $\mathcal{A}_\exists\,(\mathcal{Q}_1, \mathcal{I}_1, \mathcal{R}_1, \Phi_1)$ where Φ_1 belongs to the following logic $\mathsf{TL_{PE}}$:

Definition 4 (The Linear Time Fragment of $\mathsf{LeftCTL}^*$ **[26]).** *We define the logics* $\mathsf{TL_{PE}}$ *and* $\mathsf{TL_{PA}}$ *by the following grammar rules, where* $\mathsf{TL_{PE}}$ *and* $\mathsf{TL_{PA}}$ *are the sets of formulas that can be derived from the nonterminals* P_{PE} *and* P_{PA}*, respectively, where the rules of* P_{Prop} *are given in theorem 6:*

$$
\begin{array}{ll}
P_{\mathsf{PA}} ::= \mathcal{V} & P_{\mathsf{PE}} ::= \mathcal{V} \\
\quad \mid \neg P_{\mathsf{PE}} \mid P_{\mathsf{PA}} \wedge P_{\mathsf{PA}} \mid P_{\mathsf{PA}} \vee P_{\mathsf{PA}} & \quad \mid \neg P_{\mathsf{PA}} \mid P_{\mathsf{PE}} \wedge P_{\mathsf{PE}} \mid P_{\mathsf{PE}} \vee P_{\mathsf{PE}} \\
\quad \mid \mathsf{X} P_{\mathsf{PA}} \mid \mathsf{G} P_{\mathsf{PA}} \mid \mathsf{F} P_{\mathsf{Prop}} & \quad \mid \mathsf{X} P_{\mathsf{PE}} \mid \mathsf{G} P_{\mathsf{Prop}} \mid \mathsf{F} P_{\mathsf{PE}} \\
\quad \mid [P_{\mathsf{PA}}\ \underline{\mathsf{U}}\ P_{\mathsf{Prop}}] \mid [P_{\mathsf{PA}}\ \mathsf{U}\ P_{\mathsf{Prop}}] & \quad \mid [P_{\mathsf{Prop}}\ \underline{\mathsf{U}}\ P_{\mathsf{PE}}] \mid [P_{\mathsf{Prop}}\ \mathsf{U}\ P_{\mathsf{PE}}]
\end{array}
$$

Obviously, we have $\mathsf{TL_{PE}} \subseteq \mathsf{TL_{FG}}$ and $\mathsf{TL_{PA}} \subseteq \mathsf{TL_{GF}}$, and both inclusions are strict. It is therefore possible to modify our function BorelFG to a new procedure TopPE that does not abbreviate all temporal operators until a propositional formula is obtained, but only those that violate the grammar rules of $\mathsf{TL_{PE}}$ and $\mathsf{TL_{PA}}$, respectively.

The reason why we want to extract the largest $\mathsf{TL_{PE}}$ template of a $\mathsf{TL_{FG}}$ formula is that in [26], it has been shown that all formulas of the form $\mathsf{E}\Phi$ with $\Phi \in \mathsf{TL_{PE}}$ can be

translated to CTL. Note further that the reachability constraints generated by BorelFG do also belong to $\mathsf{TL}_{\mathsf{PE}}$. Therefore, after the application of either BorelFG or $\mathsf{TL}_{\mathsf{PE}}$, we have the choice to either translate by means of closure theorems or by a translation to CTL. This is controlled by the remaining flag *toCTL*.

7 Experimental Results

The presented translation procedures have been implemented in ML as a plug-in for the theorem prover HOL [16] and as a Java Applet that can be accessed[8] via a WWW browser with a Java virtual machine. Some experiments have already been made that have clearly shown that the presented translation procedures often outperform existing tools. To illustrate this with a practical example, we consider the verification of the arbitration process given in [30]. The specification we verified for this arbitration process is the following property that assures that no process is indefinitely ignored [30]:

$$\underbrace{\mathsf{G}\left[\left(\bigvee_{j=1}^{n}\alpha_j\right)\rightarrow \mathsf{XF}f_A\right]}_{\text{termination of access}} \wedge \underbrace{\mathsf{G}\left[\bigwedge_{j=1}^{n}(\varrho_j \rightarrow [\varrho_j\ \mathsf{U}\ \alpha_j])\right]}_{\text{persistent requests}} \rightarrow \underbrace{\mathsf{G}\bigwedge_{j=1}^{n}\mathsf{F}[\varrho_j\rightarrow\alpha_j]}_{\text{fairness}}$$

We have checked this specification using different translation procedures. The memory consumptions and runtime requirements on a Pentium-III@450MHz with 256 MBytes main memory with CMU-SMV 2.4.3 as backend model checker are given in Figure 4.

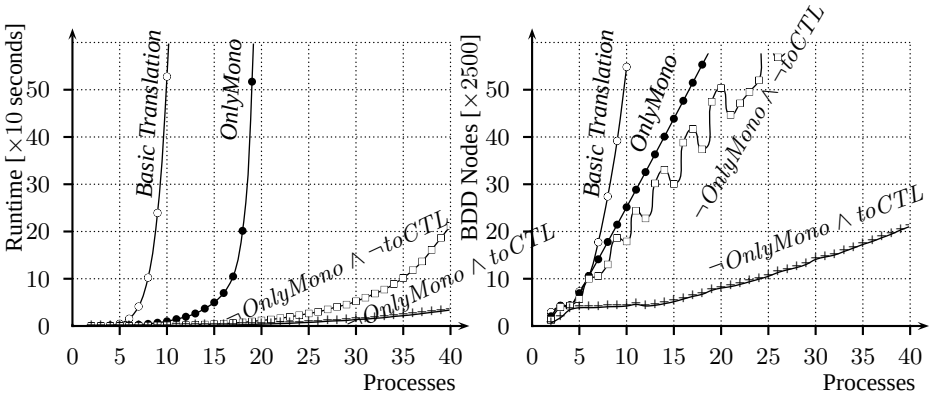


Fig. 4. Experimental data for the verification of the arbitration process

The above results have been obtained by optimized variable orderings. The basic translation introduces $2n+4$ fairness constraints for n processes, while TopProp will only generate three, and using *useBorel* even none (note that the negation of the specification is translated). Note that all the LTL model checkers of the SMV family, i.e., CMU-SMV [21, 6], CADENCE-SMV [22], and NuSMV [4] only use – beneath some other improvements, different from the ones presented here – the basic translation of theorem 3.

[8] http://goethe.ira.uka.de/∼schneider/my_tools/TempLogicTool

We have had similar experiences with other experiments (we checked over 10000 randomly generated LTL formulas with up to 100 operators). We therefore claim that our translations outperform state-of-the-art symbolic model checking tools for LTL.

References

1. R. Bloem, H. Gabow, and F. Somenzi. An algorithm for strongly connected component analysis in $n \log(n)$ symbolic steps. In *International Conference on Formal Methods in Computer Aided Design (FMCAD), LNCS 1954*, pp. 37–54. Springer Verlag, 2000.
2. R. Bloem, K. Ravi, and F. Somenzi. Efficient decision procedures for model checking of linear time logic properties. In *Conference on Computer Aided Verification (CAV), LNCS 1633*, Trento, Italy, 1999. Springer-Verlag.
3. J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computing*, 98(2):142–170, June 1992.
4. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: A new symbolic model verifier. In *Conference on Computer Aided Verification (CAV), LNCS 1633*, pp. 495–499, Trento, Italy, 1999. Springer-Verlag.
5. E. Clarke and E. Emerson. Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic. In *Workshop on Logics of Programs, LNCS 131*, pp. 52–71, Yorktown Heights, New York, May 1981. Springer-Verlag.
6. E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Conference on Computer Aided Verification (CAV), LNCS 818*, pp. 415–427, Standford, California, USA, June 1994. Springer-Verlag.
7. R. Cleaveland and B. Steffen. A linear-time model checking algorithm for the alternation-free μ-calculus. *Formal Methods in System Design*, 2(2):121–147, April 1993.
8. J.-M. Couvreur. On-the-fly verification of linear temporal logic. In *FM'99 - Formal Methods, LNCS 1708*, pp. 233–252, Toulouse, France, 1999. Springer Verlag.
9. M. Daniele, F. Giunchiglia, and M. Vardi. Improved automata generation for linear temporal logic. In *Conference on Computer Aided Verification (CAV), LNCS 1633*, Trento, Italy, 1999. Springer-Verlag.
10. E. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pp. 996–1072, Amsterdam, 1990. Elsevier Science Publishers.
11. E. Emerson and J. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.
12. E. Emerson and C.-L. Lei. Modalities for model checking: Branching time strikes back. In *ACM Symposium on Principles of Programming Languages*, pp. 84–96, New York, 1985.
13. K. Etessami and G. Holzmann. Optimizing Büchi automata. In *International Conference on Concurrency Theory, LNCS 1877*, pp. 153–168. Springer Verlag, 2000.
14. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Conference on Computer Aided Verification (CAV), LNCS 2102*, pp. 53–65, Paris, France, 2001. Springer Verlag.
15. R. Gerth, D. Peled, M. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing, and Verification (PSTV)*, Warsaw, June 1995. North-Holland.
16. M. Gordon and T. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.
17. S. Johnson, P. Miner, and A. Camilleri. Studies of the single pulser in various reasoning systems. In *International Conference on Theorem Provers in Circuit Design (TPCD), LNCS 901*, pp. 126–145, Bad Herrenalb, Germany, September 1994. Springer-Verlag.

18. Y. Kesten, A. Pnueli, and L. Raviv. Algorithmic verification of linear temporal logic specifications. In *Automata, Languages and Programming (ICALP), LNCS 1443*, Aalborg, Denmark, 1998. Springer Verlag.
19. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *ACM Symposium on Principles of Programming Languages (POPL)*, pp. 97–107, New York, January 1985. ACM.
20. Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *ACM Symposium on Principles of Distributed Computing*, pp. 377–408, 1990.
21. K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell Massachusetts, 1993.
22. K. McMillan. Cadence SMV, http://www-cad.eecs.berkeley.edu/˜kenmcmil, 2000.
23. M.Y.Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
24. A. Pnueli. The temporal logic of programs. In *Symposium on Foundations of Computer Science*, volume 18, pp. 46–57, New York, 1977. IEEE.
25. K. Ravi, R. Bloem, and F. Somenzi. A comparative study of symbolic algorithms for the computation of fair cycles. In *International Conference on Formal Methods in Computer Aided Design (FMCAD), LNCS 1954*. Springer Verlag, 2000.
26. K. Schneider. CTL and equivalent sublanguages of CTL*. In *IFIP Conference on Computer Hardware Description Languages and their Applications (CHDL)*, pp. 40–59, Toledo,Spain, April 1997. IFIP, Chapman and Hall.
27. K. Schneider. Model checking on product structures. In *Formal Methods in Computer-Aided Design, LNCS 1522*, pp. 483–500, Palo Alto, USA, November 1998. Springer Verlag.
28. K. Schneider. *Exploiting Hierarchies in Temporal Logics, Finite Automata, Arithmetics, and μ-Calculus for Efficiently Verifying Reactive Systems*. Habilitation Thesis. University of Karlsruhe, 2001.
29. K. Schneider and D. Hoffmann. A HOL conversion for translating linear time temporal logic to ω-automata. In *Higher Order Logic Theorem Proving and its Applications, LNCS 1690*, pp. 255–272, Nice, France, September 1999. Springer Verlag.
30. K. Schneider and V. Sabelfeld. Introducing mutual exclusion in Esterel. In *Andrei Ershov Third International Conference Perspectives of Systems Informatics, LNCS 1755*, pp. 445–459, Akademgorodok, Novosibirsk, Russia, July 1999. Springer Verlag.
31. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Conference on Computer Aided Verification, LNCS 1633*, pp. 247–263, Trento, Italy, 2000. Springer-Verlag.
32. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, vol. B, pp. 133–191, Amsterdam, 1990. Elsevier Science Publishers.
33. M. Vardi. Branching vs. linear time: Final showdown. In *Tools and Algorithms for the Construction and Analysis of Systems, LNCS 2031*, pp. 1–22, Genova, Italy, 2001. Springer Verlag.
34. M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *IEEE Symposium on Logic in Computer Science*, pp. 332–344. IEEE Computer Society Press, June 1986.
35. W. Visser, H. Barringer, D. Fellows, G. Gough, and A. Williams. Efficient CTL* model checking for analysis of rainbow designs. In *Conference on Correct Hardware Design and Verification Methods*, Montreal, Canada, October 1997. IFIP WG 10.5, Chapman and Hall.
36. K. Wagner. On ω-regular sets. *Information and control*, 43:123–177, 1979.
37. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.
38. P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In *Summer School on Formal Methods in Performance Analysis, LNCS 2090*, pp. 261–277. Springer Verlag, 2001.