Translating Linear Temporal Logic to Deterministic ω-Automata[1]

Klaus Schneider

University of Karlsruhe, Department of Computer Science,
Institute for Computer Design and Fault Tolerance (Prof. D. Schmid),
P.O. Box 6980, 76128 Karlsruhe, Germany,
e-mail: Klaus.Schneider@informatik.uni-karlsruhe.de
http://goethe.ira.uka.de/people/schneider

Abstract. *In this paper, a new algorithm for translating linear time temporal logic (LTL) into deterministic ω-automata is presented which circumvents the usual tableau construction. The algorithm is not limited to a special kind of ω-automaton. As most decidability problems for ω-automata can be reduced to corresponding CTL model checking problems, the presented algorithm can be used to translate LTL problems into equivalent CTL model checking problems. As a consequence, the translation method allows the use of symbolic traversal methods based on BDDs for LTL theorem proving as well as for LTL model checking.*

1 Introduction

Temporal logics are convenient formalisms for the specification of temporal behavior. In general, there are two main kinds of temporal logics: branching time temporal logics (as CTL) and linear time temporal logics (LTL) (see [1] for an excellent overview). Although the logics differ in their expressiveness [2], most useful properties can be expressed in both of them [3]. On the one hand, LTL tends to be more readable than CTL, but on the other hand, CTL model checking algorithms are less complex than those for LTL. The efficiency of CTL model checkers is to a great part based on the *symbolic* representation of sets of states. This method circumvents explicit enumeration of states by representing sets of states by their characteristic function in form of binary decision diagrams [4]. LTL model checking algorithms cannot benefit directly from this representation, since the semantics of LTL formulas is given in terms of paths through the finite state system instead of states of that system. Hence, the only way to use symbolic model checking for LTL is by translating LTL to another formalism. For this reason, LTL is usually translated by Wolper's tableau method [5] into nondeterministic Büchi automata. However, top-down proof methods as tableaux are known to be inefficient [6] in some cases, as the top-down case distinctions often overlap and hence, some cases are considered multiply.

A first approach for translating LTL into ω-automata which is not based on the tableau method has been developed in [7]. Instead, the translation method computes a deterministic Muller automaton [8] by a bottom-up traversal through the syntax tree of the LTL formula. Boolean connectives are reduced to complementation, union and intersection of the corresponding automata. The closure under temporal operators is done by first constructing an indeterministic automaton that is made deterministic afterwards. The

[1]This work has been financed by DFG project Automated System Design, SFB358.

determinization is necessary, as it is complicated to perform boolean connectives on indeterministic automata. Clearly, this approach suffers from the high complexity of the determinization algorithm which has to be applied for each temporal operator.

In this paper, a new algorithm for translating LTL theorem proving and model checking problems into deterministic ω-automata problems is presented which is neither based on the tableau method nor does it need to switch between indeterministic and deterministic automata. The translation procedure is presented in the paper with Büchi automata, but can also be used with other kinds of deterministic ω-automata, provided that these are closed under union and intersection and under two simple temporal operators. In particular, it can also be used with deterministic Rabin or Prefix automata [8, 9]. It is well-known that neither deterministic Büchi nor deterministic Prefix automata can express full LTL. For this reason, additional input variables are generated to model the nondeterminism. This means that although the generated automata have deterministic state transitions, they still have some kind of nondeterminism that arises from the additional inputs.

2 Formal Background

ω-automata are strictly stronger than LTL, but as expressive as *quantified LTL*. For this reason, quantified LTL is used as a common formalism for both ω-automata and LTL throughout the paper. The syntax of quantified LTL is given in the following definition:

Definition 1 (Syntax of QLTL_Σ) *The following mutually recursive definitions introduce the set of quantified LTL formulae $\mathcal{F}_\Sigma$ over a given finite set of variables V_Σ:*

- *each variable in V_Σ is a formula, i.e. $V_\Sigma \subseteq \mathcal{F}_\Sigma$*
- *$\mathcal{F}_\Sigma$ is closed with respect to boolean operations, i.e $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi \in \mathcal{F}_\Sigma$ if $\varphi, \psi \in \mathcal{F}_\Sigma$*
- *$\mathcal{F}_\Sigma$ is closed with respect to temporal operators, i.e $\mathsf{X}\varphi$, $\mathsf{G}\varphi$, $\mathsf{F}\varphi$, $[\varphi \;\mathsf{W}\; \psi]$, and $[\varphi \;\mathsf{U}\; \psi] \in \mathcal{F}_\Sigma$ if $\varphi, \psi \in \mathcal{F}_\Sigma$*
- *$\forall x.\varphi \in \mathcal{F}_\Sigma$ and $\exists x.\varphi \in \mathcal{F}_\Sigma$ for $x \in V_\Sigma$ and $\varphi \in \mathcal{F}_\Sigma$*

The set of LTL formulas is the set of formulas of $\mathcal{F}_\Sigma$ where no quantifier $\forall, \exists$ occurs.

The semantics is given informally as follows: $\mathsf{G}x$ holds at a certain point of time, iff x holds from this point on; $\mathsf{F}x$ holds at a certain point of time, iff x holds at least once after this point. $[x \;\mathsf{W}\; b]$ holds at a point of time t_0 iff x holds when b holds for the first time after t_0. If b never holds after t_0, then $[x \;\mathsf{W}\; b]$ holds trivially. U is the 'until'-operator, i.e. $[x \;\mathsf{U}\; b]$ holds at t_0 iff x holds until b becomes true for the first time after t_0. For each of the 'event-oriented' temporal operators W and U, one can also define strong variants $\underline{\mathsf{W}}$ and $\underline{\mathsf{U}}$ which are equivalent to the weak versions, except that they require that the event has to occur. This means $[\varphi \;\underline{\mathsf{W}}\; \psi] := [\varphi \;\mathsf{W}\; \psi] \wedge \mathsf{F}\psi$ and $[\varphi \;\underline{\mathsf{U}}\; \psi] := [\varphi \;\mathsf{U}\; \psi] \wedge \mathsf{F}\psi$.

For the description of ω-automata by quantified LTL formulas, it is assumed that the states Q and the input alphabet Σ are encoded by boolean tuples. In the following, these tuples are written as vectors $\vec{b} = (b_1, \ldots, b_n)$, and the length of a tuple $\vec{b}$ is written as $\left\|\vec{b}\right\| := n$. An infinite word $\vec{i}$ over an alphabet $\mathbb{B}^n$ is modeled as function from natural numbers $\mathbb{N}$ (that represent the time) to $\mathbb{B}^n$, where $\vec{i}^{(k)}$ is the k-th symbol in the word $\vec{i}$.

In general, the language accepted by each ω-automaton can be described by a logical formula of the form $\mathcal{A}(\vec{i}) = \exists \vec{q}. \mathcal{I}(\vec{q}) \wedge [\mathsf{G}\mathcal{T}(\vec{i},\vec{q})] \wedge \Theta(\vec{i},\vec{q})$. $\mathcal{I}(\vec{q})$ is a propositional formula that defines the set of initial states, i.e. each tuple $\vec{q}$ that satisfies $\mathcal{I}(\vec{q})$ is an initial state of the automaton $\mathcal{A}(\vec{i})$. $\mathcal{T}(\vec{i},\vec{q})$ is a propositional formula in $\vec{i}$, $\vec{q}$ and $\mathsf{X}\vec{q}$ that defines the transition relation. Given that the automaton $\mathcal{A}(\vec{i})$ is in state $\vec{q}$ and reads the input $\vec{i}$, the next state will be $\mathsf{X}\vec{q}$, provided that $\mathsf{G}\mathcal{T}(\vec{i},\vec{q})$ holds. $\Theta(\vec{i},\vec{q})$ is an LTL formula that describes the acceptance condition. A word $\vec{i}$ is accepted by the automaton iff it satisfies the formula $\mathcal{A}(\vec{i})$. In particular, Büchi automata have the following form:

Definition 2 (Büchi Automata) *Let $\Omega_k(\vec{i},\vec{q})$ for $k \in \{0,\ldots,s\}$ and $\Phi(\vec{i},\vec{q})$ be propositional formulas with the free variables $\vec{i}$ and $\vec{q}$. Moreover, let $\omega_k \in \{\mathsf{T},\mathsf{F}\}$ for $k \in \{0,\ldots,s\}$, then the following formula $\mathcal{B}(\vec{i})$ is a (deterministic) Büchi formula:*

$$\mathcal{B}(\vec{i}) := \left(\begin{array}{l} \exists q_1 \ldots q_s. \\ \quad \bigwedge_{k=0}^{s} \left[(q_k = \omega_k) \wedge \mathsf{G}\left(\mathsf{X}q_k = \Omega_k(\vec{i},\vec{q})\right)\right] \wedge \\ \quad \mathsf{GF}\Phi(\vec{i},\vec{q}) \end{array}\right)$$

An infinite word $\vec{i}$ satisfies $\mathcal{B}(\vec{i})$, iff the propositional formula $\Phi(\vec{i},\vec{q})$ holds infinitely often, when $\vec{q}$ is the corresponding run through the finite state system given by the formulae Ω_k. In the above definition, it is allowed that the acceptance condition $\Phi(\vec{i},\vec{q})$ also depends on the current input $\vec{i}$. This is not allowed in the definitions found in the literature [8]. Although this definition does not extend the expressiveness of the automaton, it makes some definitions in the following easier[2].

It is well-known that deterministic Büchi automata are not closed under complementation, but under intersection as well as union [10]:

Theorem 1 (Union and Intersection of Büchi Automata) *For arbitrary Büchi formulas $\mathcal{B}_1(\vec{i}) := \exists \vec{q}. \bigwedge_{k=0}^{s_1} \left[(q_k = \omega_{1,k}) \wedge \mathsf{G}\left(\mathsf{X}q_k = \Omega_{1,k}(\vec{i},\vec{q})\right)\right] \wedge \mathsf{GF}\Phi_1(\vec{i},\vec{q})$ and $\mathcal{B}_2(\vec{i}) := \exists \vec{p}. \bigwedge_{k=0}^{s_2} \left[(p_k = \omega_{2,k}) \wedge \mathsf{G}\left(\mathsf{X}p_k = \Omega_{2,k}(\vec{i},\vec{p})\right)\right] \wedge \mathsf{GF}\Phi_2(\vec{i},\vec{p})$, the following holds:*

- $$\mathcal{B}_1(\vec{i}) \vee \mathcal{B}_2(\vec{i}) = \left(\begin{array}{l} \exists \vec{q}. \exists \vec{p}. \\ \quad \bigwedge_{k=0}^{s_1} \left[(q_k = \omega_{1,k}) \wedge \mathsf{G}\left(\mathsf{X}q_k = \Omega_{1,k}(\vec{i},\vec{q})\right)\right] \wedge \\ \quad \bigwedge_{k=0}^{s_2} \left[(p_k = \omega_{2,k}) \wedge \mathsf{G}\left(\mathsf{X}p_k = \Omega_{2,k}(\vec{i},\vec{p})\right)\right] \wedge \\ \quad \mathsf{GF}[\Phi_1(\vec{i},\vec{q}) \vee \Phi_2(\vec{i},\vec{p})] \end{array}\right)$$
- $$\mathcal{B}_1(\vec{i}) \wedge \mathcal{B}_2(\vec{i}) = \left(\begin{array}{l} \exists r. \exists \vec{q}. \exists \vec{p}. \\ \quad \left[(r = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}r = \left(r \Rightarrow \neg\Phi_2(\vec{i},\vec{p}) \middle| \Phi_1(\vec{i},\vec{q})\right))\right] \wedge \\ \quad \bigwedge_{k=0}^{s_1} \left[(q_k = \omega_{1,k}) \wedge \mathsf{G}\left(\mathsf{X}q_k = \Omega_{1,k}(\vec{i},\vec{q})\right)\right] \wedge \\ \quad \bigwedge_{k=0}^{s_2} \left[(p_k = \omega_{2,k}) \wedge \mathsf{G}\left(\mathsf{X}p_k = \Omega_{2,k}(\vec{i},\vec{p})\right)\right] \wedge \\ \quad \mathsf{GF}[r \wedge \Phi_2(\vec{i},\vec{p})] \end{array}\right)$$

[2]While a propositional formula $\Phi(\vec{q})$ describes a set of states, a formula $\Phi(\vec{i},\vec{q})$ describes a set of edges in the transition diagram of the automaton.

As already mentioned, the translation procedure presented in this paper is not limited to Büchi automata. In [9], special ω-automata called prefix automata have been used for the same procedure (prefix automata can be viewed as the boolean closure of safety and liveness properties). The implementation of the procedure given under the URL of the author is also based on prefix automata. Nevertheless, the next section describes the procedure with deterministic Büchi automata.

3 Translating LTL into Deterministic ω-Automata

In this section, the translation method for LTL into universally quantified deterministic Büchi automata is presented. The translation of a LTL formula φ involves the following steps which are discussed in the following:

1. computation of negation normal form
2. computation of the prenex next normal form φ_p with kernel ψ_p
3. reduction of the kernel ψ_p in quantified W normal form
4. computation of the X- and W-closures and generation of certain assumptions by a bottom-up traversal through the syntax tree.

As Büchi automata are not closed under complementation, all negation symbols of the given LTL formula have to be pushed inwards over the temporal operators. This is also advantageous when other ω-automata are used, since the computation of the complement is in most cases highly complex. The transformation into negation normal form is done as follows:

Theorem 2 (Negation Normal Form) *The application of the following theorems converts each LTL formula into an equivalent one where negation symbols only occur in front of variables:*

- $\neg\neg\varphi = \varphi$, $\neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi$ *and* $\neg(\varphi \vee \psi) = \neg\varphi \wedge \neg\psi$
- $\neg\mathsf{X}\varphi = \mathsf{X}\neg\varphi$, $\neg\mathsf{G}\varphi = \mathsf{F}\neg\varphi$ *and* $\neg\mathsf{F}\varphi = \mathsf{G}\neg\varphi$
- $\neg[x\ \mathsf{W}\ b] = [(\neg x)\ \underline{\mathsf{W}}\ b] = [(\neg x)\ \mathsf{W}\ b] \wedge [\mathsf{F}b]$
- $\neg[x\ \mathsf{U}\ b] = \neg[b\ \mathsf{W}\ (x \to b)] = [(\neg b)\ \mathsf{W}\ (x \to b)] \wedge \mathsf{F}(x \to b)$

The definition and existence of the prenex next normal form is based on the following theorem. The computation itself is straightforward and is therefore not presented in detail.

Theorem 3 (Prenex Next Normal Form (PNNF)) *For every LTL formula Φ, there is an equivalent quantified LTL formula of the following form, where Ψ is a LTL formula without X-operators:*

$$\exists q_1 \ldots q_n. \left(\bigwedge_{k=1}^{n} \mathsf{G}\left(\mathsf{X}q_k = x_k\right)\right) \wedge [\mathsf{X}\ldots\mathsf{X}\Psi]$$

The variables q_k must not occur in Φ and the variables x_k are either variables occurring in Φ or one of the variables q_k. The formula Ψ is called the kernel of the above PNNF.

The X-operator commutes with all other operators, e.g. $[(\mathsf{X}y)\ \mathsf{W}\ (\mathsf{X}x)] = \mathsf{X}[y\ \mathsf{W}\ x]$ holds. However, if only one argument of a binary operator has a leading X-operator, as e.g. in $[(\mathsf{X}y)\ \mathsf{W}\ x]$, these laws cannot be used directly. In this case, a new variable q is introduced by defining $\mathsf{G}(\mathsf{X}q = x)$, such that the formula $\exists q.\mathsf{G}(\mathsf{X}q = x) \wedge [(\mathsf{X}y)\ \mathsf{W}\ (\mathsf{X}q)]$ is obtained. After that the X-operators can be shifted outwards. For example, the PNNF of $\mathsf{X}[b\ \mathsf{W}\ \mathsf{X}a] \wedge c$ is $\exists q_1 q_2 q_3.\mathsf{G}(\mathsf{X}q_1 = b) \wedge \mathsf{G}(\mathsf{X}q_2 = c) \wedge \mathsf{G}(\mathsf{X}q_3 = q_2) \wedge \mathsf{XX}\left([q_1\ \mathsf{W}\ a] \wedge q_3\right)$.

As the initial values of the new variables q_k in the above theorem are not considered for the evaluation of the truth value of Ψ, they may be arbitrarily set to T or F. The equations $\mathsf{G}(\mathsf{X}q_k = x_k)$ can be viewed as transition equations of a transition relation and hence, it is sufficient to translate the remaining formula $[\mathsf{X} \ldots \mathsf{X}\Psi]$ into an ω-automaton. The following closure theorem for the X-operator shows how this can be done, if the Ψ can be translated into a deterministic Büchi automaton:

Theorem 4 (X-Closure of Büchi Automata) *Given the Büchi automaton $\mathcal{B}(\vec{i})$ of definition 2, the following Büchi automaton is equivalent to $\mathsf{X}\mathcal{B}(\vec{i})$:*

$$\left(\begin{array}{l} \exists p\, q_1 \ldots q_s. \\ \quad [(p = \mathsf{F}) \wedge \mathsf{G}\,(\mathsf{X}p = \mathsf{T})] \wedge \\ \quad \bigwedge_{k=0}^{s} \left[(q_k = \omega_k) \wedge \mathsf{G}\left(\mathsf{X}q_k = \Omega_k(\vec{i}, \vec{q})\right)\right] \wedge \\ \quad \mathsf{GF}\Phi(\vec{i}, \vec{q}) \end{array}\right)$$

Thus, it remains to be show how LTL formulas without X-operators can be translated into deterministic Büchi automata. This is based on the following closure theorem for the temporal operator W, which can also be adapted to $\underline{\mathsf{W}}$.

Theorem 5 (W-Closure of Büchi Automata) *Given the Büchi automaton $\mathcal{B}(\vec{i})$ of definition 2 and propositional formula b, the following Büchi automaton is equivalent to $[\mathcal{B}(\vec{i})\ \mathsf{W}\ b]$:*

$$\left(\begin{array}{l} \exists p\, q_1 \ldots q_s. \\ \quad [(p = \mathsf{F}) \wedge \mathsf{G}\,(\mathsf{X}p = p \vee b)] \wedge \\ \quad \bigwedge_{k=0}^{s} \left[(q_k = \omega_k) \wedge \mathsf{G}\left(\mathsf{X}q_k = \left((b \vee p) \Rightarrow \Omega_k(\vec{i}, \vec{q}) \middle| \omega_k\right)\right)\right] \wedge \\ \quad \mathsf{GF}[q \to \Phi(\vec{i}, \vec{q})] \end{array}\right)$$

W and X form a basis, i.e. these two operators are powerful enough to express any other temporal operator, e.g. the equations $\mathsf{F}x = \neg[\mathsf{F}\ \mathsf{W}\ x]$, $\mathsf{G}x = [\mathsf{F}\ \mathsf{W}\ (\neg x)]$, and $[x\ \mathsf{U}\ b] = [b\ \mathsf{W}\ (x \to b)]$ hold. However, these equations do not allow to derive closure theorems for the remaining temporal operators from the above theorem, since the assumption that the event b has to be propositional formula is in general not fulfilled. In order to handle the closure for these operators, the following theorem is used to reduce these operators into equivalent quantified W expressions with propositional event. The key of this theorem is the introduction of new signals as events, which may become true in a special interval.

Theorem 6 (Reduction on Quantified W-expressions) *Given that the variable a does not occur in φ and ψ, the following equations hold:*

(1) $[\mathsf{G}\varphi] = \forall a.[\varphi\ \mathsf{W}\ a]$
(2) $[\mathsf{F}\varphi] = \exists a.\mathsf{F}a \wedge [\varphi\ \mathsf{W}\ a]$
(3) $[\varphi\ \mathsf{U}\ \psi] = \forall a.\left[(\neg a)\ \mathsf{U}\ \psi\right] \vee [\varphi\ \mathsf{W}\ a]$
(4) $[(\forall a.\mathcal{B}(a))\ \mathsf{W}\ \psi] = \forall a.[(\mathcal{B}(a))\ \mathsf{W}\ \psi]$
(5) $[(\exists a.\mathcal{B}(a))\ \mathsf{W}\ \psi] = \exists a.[(\mathcal{B}(a))\ \mathsf{W}\ \psi]$

```
VAL $\mathcal{V} := \{\}$;

FUNCTION add_var($\Theta$) = $\{a$ := new_var; $\mathcal{V}$ := $\mathcal{V} \cup \{(a,\Theta)\}$; return $a$; $\}$

FUNCTION QW($\varphi$) =
  CASE $\varphi$ of q
    is_prop($\varphi$)        : return PROP($\varphi$);
    $\varphi_1 \wedge \varphi_2$           : return OMEGA_CONJ(QW($\varphi_1$),QW($\varphi_2$))
    $\varphi_1 \vee \varphi_2$           : return OMEGA_DISJ(QW($\varphi_1$),QW($\varphi_2$))
    X$\varphi_1$              : return OMEGA_X(QW($\varphi_1$))
    G$\varphi_1$              : if prop($\varphi_1$) then return PROP_G($\varphi_1$)
                       else $\{a$ = add_var($\forall$); return QW($[\varphi_1$ W $a]$)$\}$
    F$\varphi_1$              : if prop($\varphi_1$) then return PROP_F($\varphi_1$)
                       else $\{a$ = add_var($\exists$); return QW(F$a \wedge [\varphi_1$ W $a]$)$\}$
    $[\varphi_1$ U $b]$         : if prop($\varphi_1$) then return PROP_U($\varphi_1, b$)
                       else $\{a$ = add_var($\forall$);
                           return QW($[(\neg a)$ U $b] \vee [\varphi_1$ W $a]$)$\}$
    $[\varphi_1$ W $b]$         : if prop($\varphi_1$) then return PROP_W($\varphi_1, b$)
                       else return OMEGA_W($b$,QW($\varphi_1$))

FUNCTION LTL2QWHEN($\varphi$) = $\{$ $\mathcal{V} := \{\}$; $\mathcal{B}$:= QW($\varphi$); return mk_quantify($\mathcal{V}$,$\mathcal{B}$); $\}$
```

Figure 1: Algorithm for translating a subset of LTL to quantified ω-automata

Equations (1), (2) in the above theorem reduce the temporal operators G, F to quantified W-expressions. If ψ is propositional equation (3), then (3) reduces U-expressions to quantified W-expressions, as $[(\neg a)\ \mathsf{U}\ \psi]$ can then be eliminated by a simple Büchi automaton (see the proof of theorem 7). Equations (4), (5) allow to shift quantifiers over W-operators such that a prenex normal form can be obtained similar to first order logic. This leads directly to the following theorem:

Theorem 7 *Given a LTL formula $\Phi(\vec{i})$ in negation normal form such that for all subformulas $[x\ \mathsf{W}\ b]$ and $[x\ \mathsf{U}\ b]$ the event b is propositional, then there is a deterministic Büchi automaton $\mathcal{B}(\vec{i},\vec{a})$ with new variables $\vec{a}$ such that for some $\Theta_j \in \{\forall,\exists\}$ the equation $\Phi(\vec{i}) = \Theta_1 a_1 \ldots \Theta_n a_{\|\vec{a}\|}.\mathcal{B}(\vec{i},\vec{a})$ holds.*

Proof: $\Theta_1 a_1 \ldots \Theta_n a_{\|\vec{a}\|}.\mathcal{B}(\vec{i},\vec{a})$ is computed by the function LTL2QWHEN of figure 1. The function OMEGA_CONJ and OMEGA_DISJ are assumed to compute the conjunction and disjunction of given ω-automata, respectively. OMEGA_X and OMEGA_W are assumed to compute the X- and W-closure of the given ω-automaton. The remaining functions are defined as follows:

- $$\mathsf{PROP}(x) := \left(\begin{array}{l} \exists pq. \\ \quad [(p = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}p = \mathsf{T})] \wedge \\ \quad [(q = \mathsf{T}) \wedge \mathsf{G}(\mathsf{X}q = q \wedge (p \vee x))] \wedge \\ \quad \mathsf{GF}q \end{array}\right)$$
- $$\mathsf{PROP\_G}(x) := \left(\begin{array}{l} \exists q. \\ \quad [(q = \mathsf{T}) \wedge \mathsf{G}(\mathsf{X}q = q \wedge x)] \wedge \\ \quad \mathsf{GF}q \end{array}\right)$$

- $\mathsf{PROP\_F}(x) := \left(\begin{array}{l} \exists q. \\ \quad [(q = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}q = q \vee x)] \wedge \\ \quad \mathsf{GF}q \end{array}\right)$
- $\mathsf{PROP\_W}(x, b) := \left(\begin{array}{l} \exists pq. \\ \quad [(p = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}p = p \vee b)] \wedge \\ \quad [(q = \mathsf{T}) \wedge \mathsf{G}(\mathsf{X}q = q \wedge (p \vee \neg b \vee x))] \wedge \\ \quad \mathsf{GF}q \end{array}\right)$
- $\mathsf{PROP\_U}(x, b) := \left(\begin{array}{l} \exists pq. \\ \quad [(p = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}p = p \vee b)] \wedge \\ \quad [(q = \mathsf{T}) \wedge \mathsf{G}(\mathsf{X}q = q \wedge (p \vee b \vee x))] \wedge \\ \quad \mathsf{GF}q \end{array}\right)$

It can be shown that the equations $\mathsf{PROP\_W}(x, b) = [x\ \mathsf{W}\ b]$, $\mathsf{PROP\_U}(x, b) = [x\ \mathsf{U}\ b]$ and $\mathsf{PROP}(x) = x$ hold. Based on these facts, the proof of the theorem is done by structural induction and follows directly the implementation of QW. The induction steps are based on the correctness of the functions OMEGA_CONJ, OMEGA_DISJ, OMEGA_X, and OMEGA_W. The induction basis follows from the correctness of the functions PROP_W, PROP_U, and PROP. ■

Hence, the closure of Büchi automata under union and intersection, X and W allows together with theorem 6 to construct for certain LTL formulas an equivalent quantified deterministic Büchi automaton. In the following, it is shown how *arbitrary* LTL formulas can be translated into such quantified automata. For this reason, the following drawbacks of the function LTL2QWHEN have to be circumvented:

1. LTL2QWHEN can only translate LTL formulas, whose events are propositional, i.e. for all subformulas $[x\ \mathsf{W}\ b]$ and $[x\ \mathsf{U}\ b]$ the event b has to be propositional.
2. LTL2QWHEN produces ω-automata with arbitrary quantification. Unfortunately, there is no 'simple' decision procedure for arbitrarily quantified ω-automata. However, as $\models \forall v_1 \ldots v_n.\mathcal{B}(\vec{i}, \vec{v})$ holds iff $\models \mathcal{B}(\vec{i}, \vec{v})$ holds, a decision procedure for (not quantified) ω-automata can also decide universally quantified ω-automata. Hence, it is the aim to modify LTL2QWHEN such that only universal quantification is produced.

Both problems are circumvented by the following trick: whenever one of the above problems is detected, the corresponding subformula φ is replaced by a new variable ℓ. Of course, it has to be provided that ℓ behaves always equivalent to φ. This is done simply by adding the assumption $\mathsf{G}(\ell = \varphi)$, according to the theorem: $\Phi(\varphi) = \forall \ell.[\mathsf{G}(\ell = \varphi)] \rightarrow \Phi(\ell)$. The detailed computation of the assumptions is given by the algorithm in figure 2 and stated in the following theorem:

Theorem 8 *Given an arbitrary LTL formula $\Phi(\vec{i})$ in negation normal form, the function* LTL2OMEGA *in figure 2 generates a set of formulas $\mathcal{E} = \{\mathsf{G}(\ell_1 = \varphi_1(\vec{i}, \vec{\ell})), \ldots, \mathsf{G}(\ell_n = \varphi_n(\vec{i}, \vec{\ell}))\}$ with new variables ℓ_j (not occurring in $\Phi(\vec{i})$) and a deterministic Büchi automaton $\mathcal{B}(\vec{i}, \vec{\ell}, \vec{a})$ such that the following holds:*

$$\Phi(\vec{i}) = \forall \ell_1 \ldots \ell_n. \bigwedge_{j=1}^{n} [\mathsf{G}(\ell_j = \varphi_j(\vec{i}, \vec{\ell}))] \rightarrow \forall a_1 \ldots a_{\|\vec{a}\|}.\mathcal{B}(\vec{i}, \vec{\ell}, \vec{a})$$

```
VAL $\mathcal{E}$ := $\{\}$;

FUNCTION add_eq($\varphi$) = $\{\ell$ = new_var; $\mathcal{E}$ := $\mathcal{E} \cup \{\mathsf{G}(\ell = \varphi)\}$; return $\ell$; $\}$

FUNCTION top($\varphi$) =
  CASE $\varphi$ of
    prop($\varphi$)          : return $\varphi$;
    $\varphi_1 \wedge \varphi_2$            : return top($\varphi_1$) $\wedge$ top($\varphi_2$);
    $\varphi_1 \vee \varphi_2$            : return top($\varphi_1$) $\vee$ top($\varphi_2$);
    X$\varphi_1$              : return add_eq(X$\mathsf{top}(\varphi_1)$);
    G$\varphi_1$              : return add_eq(G$\mathsf{top}(\varphi_1)$);
    F$\varphi_1$              : return add_eq(F$\mathsf{top}(\varphi_1)$);
    $[\varphi_1$ U $b]$          : return add_eq($[\mathsf{top}(\varphi_1)\ \mathsf{U}\ \mathsf{top}(b)]$);
    $[\varphi_1$ W $b]$          : return add_eq($[\mathsf{top}(\varphi_1)\ \mathsf{W}\ \mathsf{top}(b)]$);

FUNCTION GenFair($\varphi$) =
  CASE $\varphi$ of
    prop($\varphi$)          : return $\varphi$;
    $\varphi_1 \wedge \varphi_2$            : return GenFair($\varphi_1$) $\wedge$ GenFair($\varphi_2$);
    $\varphi_1 \vee \varphi_2$            : return GenFair($\varphi_1$) $\vee$ GenFair($\varphi_2$);
    X$\varphi_1$              : return X $(\mathsf{GenFair}(\varphi_1))$;
    G$\varphi_1$              : return G $(\mathsf{GenFair}(\varphi_1))$;
    F$\varphi_1$              : return F$(\mathsf{top}(\varphi_1))$;
    $[\varphi_1$ U $b]$          : return $[(\mathsf{GenFair}(\varphi_1))\ \mathsf{U}\ (\mathsf{top}(b))]$;
    $[\varphi_1$ W $b]$          : return $[(\mathsf{GenFair}(\varphi_1))\ \mathsf{W}\ (\mathsf{top}(b))]$;

FUNCTION LTL2OMEGA($\varphi$) =
  $\mathcal{E}$ := $\{\}$; $\psi$:= GenFair($\varphi$); $\mathcal{B}$ := LTL2QWHEN($\psi$);
  return ($\mathcal{E}$,$\mathcal{B}$);
```

Figure 2: Generation of constraints for arbitrary LTL formulas

Moreover, the formulas $\varphi_j(\vec{i}, \vec{\ell})$ *are restricted to one of the following forms* $\mathsf{G}x$, $\mathsf{F}x$, $[x\ \mathsf{U}\ b]$ *or* $[x\ \mathsf{W}\ b]$ *where both* b *and* x *are propositional.*

Proof: Given a LTL formula φ in negation normal form, the function top in figure 2 replaces all temporal subformulas of φ by new variables and adds corresponding assumptions to the set $\mathcal{E}$. For example, top($[[\mathsf{F}y]\ \mathsf{W}\ [\mathsf{G}x]] \wedge \mathsf{FG}z$) adds the assumptions $\mathsf{G}(\ell_1 = \mathsf{G}x)$, $\mathsf{G}(\ell_2 = \mathsf{F}y)$, $\mathsf{G}(\ell_3 = [\ell_2\ \mathsf{W}\ \ell_1])$, $\mathsf{G}(\ell_4 = \mathsf{G}z)$, $\mathsf{G}(\ell_5 = \mathsf{F}\ell_4)$ to $\mathcal{E}$ and finally returns $\ell_3 \wedge \ell_5$. Note that top(φ) is always propositional and note also that top(φ)=φ, if φ is propositional. top is used by the function GenFair to replace non-propositional events b of subformulas $[x\ \mathsf{U}\ b]$ and $[x\ \mathsf{W}\ b]$ by propositional ones. Thus the first of the two problems of LTL2QWHEN is circumvented. The introduction of $\exists$-quantifiers by LTL2QWHEN arises only when a subformula $\mathsf{F}\varphi_1$ is replaced by its equivalent W expression. The call of the function top in the function GenFair (in case $\varphi = \mathsf{F}\varphi_1$ eliminates this case. ■

It remains now to translate the assumptions $\mathsf{G}(\ell_j = \varphi_j)$ into ω-automata. As $\mathsf{G}x = [\mathsf{F}\ \mathsf{W}\ (\neg x)]$, $\mathsf{F}x = \neg[\mathsf{F}\ \mathsf{W}\ x]$ and $[x\ \mathsf{U}\ b] = [b\ \mathsf{W}\ (x \rightarrow b)]$ holds, it is sufficient to be able to translate $\mathsf{G}\,(\ell = [x\ \mathsf{W}\ b])$[3]:

Lemma 1 (Replacing Definitions by Büchi Automata) $\mathsf{G}\,(\ell = [x\ \mathsf{W}\ b])$ *is equivalent to the following Büchi automaton:*

$$\left(\begin{array}{l} \exists p\, q. \\ \quad (p = \mathsf{T}) \wedge \mathsf{G}\,(\mathsf{X}p = (\ell \wedge q \wedge [x \vee \neg b]) \vee (\neg\ell \wedge q \wedge b \wedge \neg x)) \wedge \\ \quad (q = \mathsf{T}) \wedge \mathsf{G}\,(\mathsf{X}q = (\neg\ell \wedge q \wedge \neg\,[x \wedge b]) \vee (\ell \wedge p \wedge b \wedge x)) \wedge \\ \quad \mathsf{GF}p \end{array}\right)$$

The above theorem has been constructed by computing the accepting Büchi automata by Wolpers tableau method. In order to finally reduce a LTL formula $\Phi(\vec{i})$ into an equivalent language inclusion problem, simply apply the function LTL2OMEGA to obtain $\forall\vec{\ell}.\ \bigwedge_{j=1}^{n}[\mathsf{G}(\ell_j = \varphi_j(\vec{i},\vec{\ell}))] \rightarrow \forall\vec{a}.\mathcal{B}(\vec{i},\vec{\ell},\vec{a})$. After that, shift the quantification $\forall\vec{a}$ outwards to $\forall\vec{\ell}$ and replace the assumptions $\mathsf{G}(\ell_j = \varphi_j(\vec{i},\vec{\ell}))$ by deterministic Büchi automata $\mathcal{B}_j(\vec{i},\vec{\ell})$ according to the last theorem. After the conjunction on the automata $\mathcal{B}_j(\vec{i},\vec{\ell})$ is performed by theorem 1 to obtain a single Büchi automaton $\mathcal{B}_{\mathcal{E}}(\vec{i},\vec{\ell})$, the given formula is finally translated to: $\forall\vec{\ell}\forall\vec{a}.\mathcal{B}_{\mathcal{E}}(\vec{i},\vec{\ell}) \rightarrow \mathcal{B}(\vec{i},\vec{\ell},\vec{a})$. *Hence, the procedure can translate arbitrary LTL problems into equivalent language inclusion problems with additional input variables $\vec{\ell}$ and $\vec{a}$.*

As an example, consider the formula $[(\mathsf{F}x)\ \mathsf{U}\ (\mathsf{G}y)]$. The function LTL2_OMEGA will generate the fairness constraint $\mathsf{G}(\ell = \mathsf{G}y)$ and consider the formula $[(\mathsf{F}x)\ \mathsf{U}\ \ell]$ afterwards, where the event of the U-operator is now propositional. As the lefthand argument of the U-operator is not propositional, the function PROP_U can not be applied in the function LTL2_QWHEN. Instead, the expression is reduced to the quantified W-expression $\forall a.\ [(\neg a)\ \mathsf{U}\ \ell] \vee [(\mathsf{F}x)\ \mathsf{W}\ a]$. Hence, $[(\mathsf{F}x)\ \mathsf{U}\ (\mathsf{G}y)]$ is equivalent to:

$$\forall\ell.\forall a.[\mathsf{G}(\ell = \mathsf{G}y)] \rightarrow [(\neg a)\ \mathsf{U}\ \ell] \vee [(\mathsf{F}x)\ \mathsf{W}\ a]$$

$\mathsf{F}x$ is replaced by the function PROP_F by $\exists s.[(s = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}s = s \vee x)] \wedge \mathsf{GF}s$. The W-closure of the latter Büchi automaton is then

$$\left(\begin{array}{l} \exists r\ s. \\ \quad [(r = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}r = r \vee a)] \wedge \\ \quad [(s = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}s = s \vee x)] \wedge \\ \quad \mathsf{GF}(r \rightarrow s) \end{array}\right)$$

The remaining U-expression can now be replaced by the function PROP_U and the disjunction of the obtained Büchi automaton with the above one is then

$$\left(\begin{array}{l} \exists p\ q\ r\ s. \\ \quad [(p = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}p = p \vee \ell)] \wedge \\ \quad [(q = \mathsf{T}) \wedge \mathsf{G}(\mathsf{X}q = q \wedge (p \vee \ell \vee \neg a))] \wedge \\ \quad [(r = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}r = r \vee a)] \wedge \\ \quad [(s = \mathsf{F}) \wedge \mathsf{G}(\mathsf{X}s = s \vee x)] \wedge \\ \quad \mathsf{GF}[q \vee (r \rightarrow s)] \end{array}\right)$$

[3]In case of a definition $\mathsf{G}(\ell = \mathsf{F}x)$, ℓ is substituted by $\neg\ell_1$, where ℓ_1 is also a new variable, and then replaced by $\mathsf{G}(\ell_1 = \mathsf{G}\neg x)$.

Finally, the assumption $\mathsf{G}(\ell = \mathsf{G}y)$ is replaced by a Büchi automaton according to lemma 1, such that the validity problem of the given LTL formula has been reduced to an equivalent language inclusion problem of two deterministic Büchi automata.

4 Conclusions and Future Work

In this paper, a new method has been presented for translating LTL theorem proving and LTL model checking problems into deterministic ω-automata problems which can in turn be expressed as CTL model checking problems. As a result, the powerful symbolic traversal methods which use BDDs for representing sets of states can also be used for solving LTL theorem proving and model checking problems. The translation method has been presented with Büchi automata, but it is possible to adapt the method such that other kinds of ω-automata that are closed under $\wedge$, $\vee$ and under the operators X and W are used.

References

[1] E.A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072, Amsterdam, 1990. Elsevier Science Publishers.

[2] E.A. Emerson and J.Y. Halpern. ”sometimes” and ”not never” revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.

[3] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 97–107, New York, January 1985. ACM.

[4] R.E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[5] P. Wolper. On the relation of programs and computations to models of temporal logic. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, pages 75–123, Altrincham, UK, 1987. Springer-Verlag.

[6] M. d’Agostino. Are Tableaux an Improvement of Truth-Tables? Cut-Free Proofs and Bivalence. *Journal of Logic, Language, and Information*, 1(3):127–139, 1992.

[7] G.G de Jong. An automata theoretic approach to temporal logic. In K.G. Larsen and A. Skou, editors, *Proceedings of 3^{rd} Workshop on Computer Aided Verification (CAV91)*, volume 575 of *Lecture Notes in Computer Science*, pages 477–487, Aalborg, July 1991. Springer-Verlag.

[8] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191, Amsterdam, 1990. Elsevier Science Publishers.

[9] K. Schneider. Translating LTL model checking to ctl model checking. Technical Report SFB358-C2-3/96, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1996.

[10] D. Siefkes. *Decidable Theories I: Büchi’s Monadic Second Order Successor Arithmetic*. Lecture Notes in Mathematics. Springer-Verlag, 1970.