Yet Another Look at LTL Model Checking*

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
Klaus.Schneider@informatik.uni-karlsruhe.de
http://goethe.ira.uka.de/~schneider

Abstract. A subset of LTL is presented that can be translated to ω-automata with only a linear number of states. The translation is completely based on closures under temporal and boolean operators. Moreover, it is shown how this enhancement can be combined with traditional translations so that all LTL formulas can be translated. Exponential savings are possible in terms of reachable states, as well as in terms of runtime and memory requirements for model checking.

1 Introduction

Nearly all decision procedures for the linear-time temporal logic LTL are based on a translation to equivalent ω-automata. Given an LTL formula Φ, the states of the corresponding ω-automaton $\mathcal{A}_\Phi$ are usually given as the powerset of the elementary formulas, i.e., of the set of all subformulas of Φ that start with a temporal logic operator. The transition relation and the acceptance condition of the automaton is determined by the fixpoint characterization of the formulas in the corresponding state (cf. [6,1]).

Clarke, Grumberg and Hamaguchi [2] pointed out that for LTL model checking, there is no reason to construct the automaton $\mathcal{A}_\Phi$ explicitely. Instead, they directly abbreviated each elementary subformula φ of Φ by a new state variable ℓ_φ. The transition relation and the acceptance condition are then directly given in terms of these state variables. This yields in a translation procedure that runs in time $O(|\Phi|)$ and whose result can be directly used for symbolic model checking.

The number of possible states is however of order $O(2^{|\Phi|})$ since any elementary formula of Φ may double the set of reachable states. Although this exponential blow-up can not be circumvented in general, it can be avoided for many formulas, when the automaton is derived by means of closures (see [6] for an example): Given that we have already derived an automaton $\mathcal{A}_\varphi$ for φ, and we have a propositional formula ψ, the presented closure theorems tell us how to construct automata for $\mathsf{X}\varphi$, $[\psi\ \underline{\mathsf{U}}\ \varphi]$, and $[\varphi\ \mathsf{B}\ \psi]$ by only introducing one additional state.

* This work has been financed by the DFG project 'Verification of embedded systems' in the priority program 'Design and design methodology of embedded systems'.

L. Pierre and T. Kropf (Eds.): CHARME'99, LNCS 1703, pp. 321–326, 1999.

A forerunner of this approach is due to Jong [3] who already used closure theorems for X and F. We extend his idea to *all* temporal operators, i.e. also to the binary ones. The closures are used to keep the automaton as small as possible by avoiding the introduction of too much states and acceptance constraints. Our translation can still be performed within $O(|\Phi|)$ time and still computes the transition relation of the automaton $\mathcal{A}_\varphi$ in a symbolic manner similar to [2].

However, the mentioned closures can only be applied if ψ is propositional, so that there is an intimate relationship between the logic LeftCTL* [5,6] and the closure theorems. LeftCTL* is a branching-time temporal logic that is equal expressive as CTL, but has a much richer syntax, so that a lot of LTL formulas that are equivalent to some CTL formulas do even syntactically belong to LeftCTL*. Using the closure theorems, we will see that any formula of LTL ∩ CTL can be translated to a deterministic Büchi automaton. A similar result has been obtained by Kupferman and Vardi, who showed in [7] that each formula of the intersection of LTL with the alternation-free μ-calculus (a superset of CTL in some sense) can be translated to a deterministic Büchi automaton.

In [6], it has been shown how given CTL* model checking problems can be handled by the extraction of a LeftCTL* formula. It has already been remarked there that the procedure presented in [6] is a generalization of the usual translation of LTL to ω-automata (in the sense of [1]). Our overall approach is therefore as follows: We apply the 'extraction procedure' of [6] to a given LTL formula Φ. This essentially computes an ω-automaton $\mathcal{A}_\Phi$ for the parts of Φ that do not belong to LeftCTL* and some formula $\Psi \in \mathsf{LTL} \cap \mathsf{LeftCTL}^*$. Ψ is then translated by means of the closure theorems to another ω-automaton $\mathcal{A}_\Psi$, so that the resulting automaton $\mathcal{A}_\Phi \times \mathcal{A}_\Psi$ is equivalent to Φ. The construction of $\mathcal{A}_\Psi$ does not introduce any fairness constraints at all, whereas the traditional translation generates $O(|\Phi|)$ fairness constraints that evidently increase the runtime requirements of the later model checking. Due to lack of space, the paper is not self-contained. It refers to some definitions and algorithms given in [5,6,1] (available from http://goethe.ira.uka.de/~schneider).

2 Translating LeftCTL* ∩ LTL to ω-Automata by Closures

We describe ω-automata by formulas of the form $\mathcal{A}_\exists(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi_\mathcal{F})$, where Q is the set of state variables (states of the automaton are given as subsets of Q). $\Phi_\mathcal{I}$ is a propositional formula over Q and describes the initial states (all sets that 'satisfy' $\Phi_\mathcal{I}$). Similarly, $\Phi_\mathcal{R}$ describes the transition relation, and $\Phi_\mathcal{F}$ the acceptance condition. In particular, we consider the following types of acceptance conditions, where Φ, Φ_k, Ψ_k are propositional over Q, and hence denote sets of states ($\mathfrak{B}^+(\varphi_1, \ldots, \varphi_n)$ denotes the set of formulas that can be constructed with the boolean connectives $\wedge$ and $\vee$ from the formulas $\varphi_1, \ldots, \varphi_n$):

Büchi Acceptance:	$\mathsf{GF}\Phi$
Generalized Büchi Acceptance:	$\mathfrak{B}^+(\mathsf{GF}\Phi_1, \ldots, \mathsf{GF}\Phi_m)$
Generalized Co-Büchi Acceptance:	$\mathfrak{B}^+(\mathsf{FG}\Phi_1, \ldots, \mathsf{FG}\Phi_m)$

It is well-known that generalized co-Büchi automata are less expressive than Büchi automata. The latter are closed under all boolean operations, but the former are not closed under complement. Moreover, generalized co-Büchi automata can be made deterministic (with an exponential blow-up) [8], while Büchi automata can not be made deterministic [4].

We next show how the LTL formulas that are also LeftCTL* [5] formulas can be translated to generalized nondeterministic co-Büchi automata with only $O(|\Phi|)$ states and an acceptance condition of length $O(|\Phi|)$. The grammar rules of LeftCTL* are as follows (see [5] for the semantics):

$$\begin{aligned} S ::= {} & \mathsf{Variables} \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P_E \mid \mathsf{A}P_A \\ P_E ::= {} & S \mid \neg P_A \mid P_E \wedge P_E \mid P_E \vee P_E \mid \mathsf{X}P_E \mid \mathsf{G}S \mid \mathsf{F}P_E \\ & \mid [P_E \,\mathsf{W}\, S] \mid [S \,\mathsf{U}\, P_E] \mid [P_E \,\mathsf{B}\, S] \mid [P_E \,\underline{\mathsf{W}}\, S] \mid [S \,\underline{\mathsf{U}}\, P_E] \mid [P_E \,\underline{\mathsf{B}}\, S] \\ P_A ::= {} & S \mid \neg P_E \mid P_A \wedge P_A \mid P_A \vee P_A \mid \mathsf{X}P_A \mid \mathsf{G}P_A \mid \mathsf{F}S \\ & \mid [P_A \,\mathsf{W}\, S] \mid [P_A \,\mathsf{U}\, S] \mid [S \,\mathsf{B}\, P_E] \mid [P_A \,\underline{\mathsf{W}}\, S] \mid [P_A \,\underline{\mathsf{U}}\, S] \mid [S \,\underline{\mathsf{B}}\, P_E] \end{aligned}$$

We call a formula without path quantifiers A and E a P_E-formula if it can be derived from the nonterminal P_E with the above grammar rules. The translation of these P_E-formulas to ω-automata by closures is done by the following equations (any LeftCTL* formula can be reduced to one with only the operators X, $\underline{\mathsf{U}}$ and B):

Lemma 1 (Temporal Closure of Generalized Co-Büchi Automata). *Given a formula Φ of $\mathfrak{B}^+(\mathsf{FG}\Phi_1, \ldots, \mathsf{FG}\Phi_n)$ and a variable p. Define for any set Q of variables the formula $\Phi_{\overline{Q}} := \bigwedge_{q \in Q} \neg p$. Then, the following equations are valid:*

- *for propositional φ, we have* $\varphi = \mathcal{A}_\exists \left(\begin{array}{l} \{p, q\}, \neg p \wedge q, \\ (\mathsf{X}p) \wedge (\mathsf{X}q = q \wedge (p \vee \varphi)), \\ \mathsf{FG}q \end{array} \right)$
- $\mathsf{X}\left[\mathcal{A}_\exists\left(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi\right)\right] = \mathcal{A}_\exists \left(\begin{array}{l} Q \cup \{p\}, \neg p \wedge \Phi_{\overline{Q}}, \\ \left[\neg p \wedge \Phi_{\overline{Q}} \wedge \mathsf{X}p \wedge \mathsf{X}\Phi_\mathcal{I}\right] \vee \left[p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p\right], \\ \Phi \end{array} \right)$
- $\left[\varphi \,\underline{\mathsf{U}}\, \mathcal{A}_\exists\left(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi\right)\right] = \mathcal{A}_\exists \left(\begin{array}{l} Q \cup \{p\}, \neg p \wedge \Phi_{\overline{Q}}, \\ \left[\neg p \wedge \Phi_{\overline{Q}} \wedge \varphi \wedge \mathsf{X}(\neg p \wedge \Phi_{\overline{Q}})\right] \vee \\ \left[\neg p \wedge \Phi_{\overline{Q}} \wedge \varphi \wedge \mathsf{X}(p \wedge \Phi_\mathcal{I})\right] \vee \\ \left[p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p\right], \\ \Phi \end{array} \right)$
- $\left[\mathcal{A}_\exists\left(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi\right) \;\mathsf{B}\; \varphi\right] = \mathcal{A}_\exists \left(\begin{array}{l} Q \cup \{p\}, \neg p \wedge \Phi_{\overline{Q}}, \\ \left[\neg p \wedge \Phi_{\overline{Q}} \wedge \neg\varphi \wedge \mathsf{X}(\neg p \wedge \Phi_{\overline{Q}})\right] \vee \\ \left[\neg p \wedge \Phi_{\overline{Q}} \wedge \neg\varphi \wedge \mathsf{X}(p \wedge \Phi_\mathcal{I} \wedge \neg\varphi)\right] \vee \\ \left[p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p\right], \\ \mathsf{FG}(\neg p \wedge \Phi_{\overline{Q}}) \vee \Phi \end{array} \right)$

Instead of giving a formal proof, we only explain the intuition that is behind the closure under $\underline{\mathsf{U}}$: a new initial state (encoded by $\neg p \wedge \Phi_{\overline{Q}}$) is added with a self-loop (encoded by $\neg p \wedge \Phi_{\overline{Q}} \wedge \varphi \wedge \mathsf{X}(\neg p \wedge \Phi_{\overline{Q}})$) that is enabled under φ. Also, there are transitions (encoded by $\neg p \wedge \Phi_{\overline{Q}} \wedge \varphi \wedge \mathsf{X}(p \wedge \Phi_\mathcal{I})$) to any initial

state $\Phi_{\mathcal{I}}$ that are also enabled under φ. If p has become true, it will stay for all the future time true and therefore enables the transitions $\Phi_{\mathcal{R}}$ of the (sub)automaton $\mathcal{A}_{\exists}(Q, \Phi_{\mathcal{I}}, \Phi_{\mathcal{R}}, \Phi)$ (encoded by $p \wedge \Phi_{\mathcal{R}} \wedge \mathsf{X}p$). The acceptance condition Φ requires that we can not stay forever in the added state $\neg p \wedge \Phi_{\overline{Q}}$, which must however be allowed for the weak U operator (and also for the B operator). The closures under X and B are constructed in a similar manner.

```
function extract_P_E(φ)
  case φ of
    is_prop(φ)      : return ({}, φ);
    φ1 ∧ φ2         : (E1, ψ1) ≡ extract_P_E(φ1); (E2, ψ2) ≡ extract_P_E(φ2);
                      return (E1 ∪ E2, ψ1 ∧ ψ2);
    φ1 ∨ φ2         : (E1, ψ1) ≡ extract_P_E(φ1); (E2, ψ2) ≡ extract_P_E(φ2);
                      return (E1 ∪ E2, ψ1 ∨ ψ2);
    Xφ1             : (E1, ψ1) ≡ extract_P_E(φ1); return (E1, Xψ1);
    [φ1 B φ2]       : (E1, ψ1) ≡ extract_P_E(φ1); (E2, ψ2) ≡ tableau(φ2);
                      return (E1 ∪ E2, [ψ1 B ψ2]);
    [φ1 U φ2]       : (E1, ψ1) ≡ tableau(φ1); (E2, ψ2) ≡ extract_P_E(φ2);
                      return (E1 ∪ E2, [ψ1 U ψ2]);

function close(φ)
  case φ of q
    is_prop(φ)      : return Π_prop(φ);
    φ1 ∧ φ2         : return Ω∧(close(φ1), close(φ2));
    φ1 ∨ φ2         : return Ω∨(close(φ1), close(φ2));
    Xφ1             : return ΩX(close(φ1));
    [φ1 B b]        : ΩB(b, close(φ1))
    [φ1 U b]        : ΩU(b, close(φ1))

function Closed_Tableau(Φ)
  ({ℓ1 = φ1, ..., ℓn = φn}, Ψ) := extract_P_E(NNF(Φ));
  Φ_R := ⋀_{i=1}^{n} trans(ℓi, φi);
  Φ_F := ⋀_{i=1}^{n} fair(ℓi, φi);
  A_∃(Q, Ψ_I, Ψ_R, Ψ_F) := close(Ψ);
  return A_∃(Q ∪ {ℓ1, ..., ℓb}, Φ_I ∧ Ψ_I, Φ_R ∧ Ψ_R, Φ_F ∧ Ψ_F);
```

Fig. 1. Algorithm for translating LTL to ω-automata by means of closures

Note that only one further state is added by the above closures (we use a wasteful encoding that introduces a new state variable for the encoding of the new state). Similar closures hold also for generalized Büchi automata, and also for conjunction and disjunction. Assume that the functions Ω_{X}, Ω_{U}, Ω_{B}, $\Omega_{\wedge}$, $\Omega_{\vee}$ implement the closures under X, U, B, $\wedge$, and $\vee$, respectively. Π_{prop} computes the ω-automaton for propositional formulas. Using these functions, the function call $\mathsf{close}(\Phi)$ with a P_E-formula Φ for the algorithm given in figure 1 yields in a nondeterministic generalized co-Büchi automaton $\mathcal{A}_{\exists}(Q, \Phi_{\mathcal{I}}, \Phi_{\mathcal{R}}, \Phi_{\mathcal{F}})$ that is

equivalent to Φ. This automaton has $O(|\Phi|)$ states and an acceptance condition of length $O(|\Phi|)$.

In [8], it is shown that generalized co-Büchi automata can be made deterministic, and that we can reduce the acceptance condition to the normal form $\bigvee_{i=0}^{f} \mathsf{FG}\Phi_i$ where even $f = 0$ will do. Hence, we can compute for any $\mathsf{LeftCTL}^*$ formula a deterministic co-Büchi automaton. As P_A is dual to P_E, and deterministic Büchi automata are the complements of deterministic co-Büchi automata, this means that we can compute for any P_A formula a deterministic Büchi automaton only by application of the above closure theorems. Hence, we have the following result (similar to [7]).

Theorem 2. *The following facts are valid:*

1. *For all $\Phi \in P_E$ there is an equivalent nondeterministic generalized co-Büchi automaton with $O(|\Phi|)$ states and an acceptance condition of length $O(|\Phi|)$.*
2. *For all $\Phi \in P_E$ there is an equivalent nondeterministic ω-automaton with $O(|\Phi|)$ states and an acceptance condition of the form $\bigvee_{i=1}^{n} \mathsf{FG}\varphi_i$ with propositional φ_i, $n \in 2^{O(|\Phi|)}$, and $|\varphi_i| \in O(|\Phi|)$.*
3. *For all $\Phi \in P_E$ there is an equivalent deterministic co-Büchi automaton.*
4. *For all $\Phi \in P_A$ there is an equivalent deterministic Büchi automaton.*

3 Translating LTL to ω-Automata by Closures

To translate an arbitrary LTL formula Φ, we abbreviate each subformula φ of Φ that violates the grammar rules of P_E by a new variable ℓ_φ (see function $\mathsf{extract\_}P_E(\varphi)$ of figure 1). The resulting P_E-formula Ψ can then be translated by means of the closures to a nondeterministic generalized co-Büchi automaton $\mathcal{A}_\Psi$. Of course, it has to be assured that ℓ behaves always equivalent to φ. According to the product model checking approach [6], we therefore add transitions and fairness constraints according to [6] (see also [1] for the functions trans and fair). Hence, we obtain the following result:

Theorem 3. *For any structure $\mathcal{K}$, any state s of $\mathcal{K}$, and any quantifier-free formula Φ, the following is equivalent for $\mathfrak{A} := \mathcal{A}_\exists(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi_\mathcal{F}) = \mathsf{Closed\_Tableau}(\Phi)$ ($\mathcal{K}_\mathfrak{A}$ is the Kripke structure associated with $\mathfrak{A}$):*

- *there is a fair path π starting in s such that $(\mathcal{K}, \pi) \models \Phi$ holds*
- *there is an initial state s_0 of $\mathfrak{A}$ such that there is a fair path $\pi \times \pi_\mathfrak{A}$ through $\mathcal{K} \times \mathcal{K}_\mathfrak{A}$ such that $(\mathcal{K} \times \mathcal{K}_\mathfrak{A}, (s, s_0)) \models \Phi_\mathcal{F}$ holds.*

Moreover, $\Phi_\mathcal{F}$ is of the form $(\bigwedge_{i=1}^{n} \mathsf{GF}\Phi_i) \wedge \Psi$, where Ψ is built-up with conjunctions and disjunctions of $\mathsf{FG}\Psi_i$ formulas.

There is also the possibility to abbreviate more subformulas than necessary (for extracting a P_E formula), so that one can gradually choose between the traditional translation and the one based on closures for optimizations.

References

1. 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, Nice, France, September 1999. Springer Verlag. 321, 322, 325
2. E. M. 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. 321, 322
3. G.G de Jong. An automata theoretic approach to temporal logic. In *Computer Aided Verification (CAV)*, LNCS 575, pp. 477–487, Aalborg, July 1991. Springer-Verlag. 322
4. 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. 323
5. K. Schneider. CTL and equivalent sublanguages of CTL*. In C. Delgado Kloos, editor, *IFIP Conference on Computer Hardware Description Languages and their Applications (CHDL)*, pp. 40–59, Toledo, Spain, April 1997. IFIP, Chapman and Hall. 322, 323
6. K. Schneider. Model checking on product structures. In *Formal Methods in Computer-Aided Design*, LNCS 1522, pp. 483–500, Palo Alto, CA, November 1998. Springer Verlag. 321, 322, 325
7. O. Kupferman and M. Y. Vardi. Freedom, weakness, and determinism: From linear-time to branching-time. In *IEEE Symposium on Logic in Computer Science*, 1998. 322, 325
8. K. Wagner. On ω regular sets. *Information and control*, 43:123–177, 1979. 323, 325