Constructing Büchi Automata from Linear Temporal Logic Using Simulation Relations for Alternating Büchi Automata*

Carsten Fritz

Institut für Informatik und Praktische Mathematik, CAU Kiel, Germany
fritz@ti.informatik.uni-kiel.de

Abstract. We present a new procedure for the translation of propositional linear-time temporal logic (LTL) formulas to equivalent nondeterministic Büchi automata. Our procedure is based on simulation relations for alternating Büchi automata. Whereas most of the procedures that have been described in the past compute simulation relations in the last step of the translation (after a nondeterministic Büchi automaton has already been constructed), our procedure computes simulation relations for alternating Büchi automata in an early stage and uses them in an on-the-fly fashion. This decreases the time and space consumption without sacrificing the potential of simulation relations.
We present experimental results that demonstrate the advantages of our approach: Our procedure is faster than TMP but produces, on the average, automata of about the same size; LTL2BA is faster than our procedure but produces larger automata.

1 Introduction

Propositional linear-time temporal logic (LTL for short) is a popular language for the specification of system properties. The standard way of model checking an LTL spec against a system is to translate the negation of the spec into an equivalent nondeterministic Büchi automaton (which incurs an exponential blow-up), build the product of this automaton with the system, and check this product for emptiness—it is empty if and only if the system satisfies the spec.

Obviously, the size of the Büchi automaton for the LTL formula is crucial for the efficiency of the above procedure. But minimizing Büchi automata is computationally difficult: Even testing universality for nondeterministic finite automata on finite strings is PSPACE-hard [GJ79]. This implies that approximating a minimum-size ω-automaton (up to a constant factor) is impossible in polynomial time unless P = PSPACE.

In practice, various heuristics are in use for state-space reduction of the resulting automata. Standard techniques are simplifications of the input formula using a set of rewrite rules [MP92], and modifications in the transition structure of the resulting Büchi automaton (cf. [EH00]). Quotienting with respect to

* Supported by Deutsche Forschungsgemeinschaft under project no. 223228.

O.H. Ibarra and Z. Dang (Eds.): CIAA 2003, LNCS 2759, pp. 35–48, 2003.

simulation or bisimulation equivalences is a sophisticated example for the latter [DHW91, ESW01, GBS02]. In general, a transition system simulates another system if, for every computation in the simulated system, there is a matching (w. r. t. an acceptance condition) computation in the simulating system. That is, a preorder on the states of the automaton is computed such that (usually) equivalent states can be merged.

Our algorithm which we present in this paper is also based on simulation relations, but our approach is different in that we compute a simulation relation *before* the exponential blow-up (see above) occurs.

The algorithm. In order to save time and space, we do not compute a simulation relation for the nondeterministic Büchi automaton. In our algorithm, a so-called *delayed* simulation relation [ESW01] is computed for an intermediate *alternating* Büchi automaton. The intermediate automaton can be interpreted as just another way of writing down the LTL formula—especially, the alternating automaton is only linear in the length of the formula. Consequently, the computation of the relation is fast in comparison to other simulation-based approaches (in the best case, exponentially faster).

In the other approaches, the crucial step is to actually compute a simulation quotient; in the procedure presented here, quotienting of the alternating automaton is not a necessary step of the construction. Instead, we use the simulation relation for on-the-fly simplifications in the computation of the result, thus again speeding up the process and saving memory resources. The price of this of course is that the resulting automaton may still contain simulation equivalent states, but our experiments indicate that this drawback is compensated by the advantage of using alternating automata in an intermediate stage.

Our construction proceeds in three main steps. First, the LTL formula is translated, in a very direct way, to an alternating Büchi automaton in which every state has either a universal or existential modality and in which we allow transitions labeled with the empty word "ε" (Section 3). We then compute the delayed simulation relation on the states of this automaton, using a variant of the game rules of [FW02b] that takes ε-transitions into account (Section 4). In the third step, we translate the alternating automaton to a nondeterministic (i. e., non-alternating) Büchi automaton, using the method of [MH84]. In this translation, we use the simulation relation for on-the-fly simplifications (Section 5). (In a fourth step in the implementation, we also do some a-posteriori simplifications not based on simulation relations.)

One advantage of our approach is that other techniques can easily be integrated into the algorithm. To give a clearer picture of the key aspects of our procedure, we did not integrate simplifications of the input formula into our algorithm (aside from computing the formula's negation normal form), but formula rewriting can improve the result.

In Section 6, we report experimental comparisons of a prototypical implementation [FT] of our algorithm with the programs LTL2BA [Odd] and TMP [Ete], using a tool of Tauriainen and Heljanko [TH02]. Our experiments show that the automata produced by our implementation are, on the average, as good (i. e., as

small) as the automata of TMP (the automata of LTL2BA are larger). But with the complexity of the formulas increasing, our program becomes substantially faster than TMP, while LTL2BA is the fastest of the three programs.

Related work. The theoretical foundations of our algorithm are mainly taken from [FW02b], but only a restricted framework (letters instead of propositions, no ε-transitions) is considered there. The game rules of Arnold and Santocanale [AS03] are quite similar to the rules we use here.

Translators from LTL to Büchi automata include an implementation as part of Holzmann's model checker Spin [Hol], Etessami's "Temporal Massage Parlor" (TMP) [Ete], LTL2BA [Odd] of Gastin and Oddoux, and "Wring" [Blo] by Somenzi and Bloem.

Both the Spin algorithm of Etessami and Holzmann [EH00] and TMP are based on an algorithm of Gerth, Peled, Vardi, and Wolper [GPVW95]. The Spin algorithm uses direct simulation while the latest version of TMP uses delayed simulation quotienting [ESW01] for the simplification of the resulting automaton. The algorithm of LTL2BA [GO01] mainly relies on a set of simple yet efficient modifications of intermediate (alternating and generalized) Büchi automata. The Wring algorithm [SB00] also uses direct simulation quotienting. Note that the output of Wring is a generalized Büchi lautomaton with propositional labels on the states, while the other programs (including our implementation) produce transition-labeled Büchi automata, so a real comparison to Wring is not possible.

For simulation in general, the reader is referred to [Mil71, HHK95]. Bisimulation was introduced by Milner and Park [Mil80, Par81]. Henzinger, Kupferman, and Rajamani [HKR97, HR00] introduce fair simulation and bisimulation relations and give efficient algorithms for computing them. Etessami, Schuller, and Wilke [ESW01] improve on these results using a variant of an algorithm of Jurdziński [Jur00]. Gurumurthy, Bloem, and Somenzi [GBS02] study fair simulation quotienting. Alur, Henzinger, Kupferman, and Vardi [AHKV98] study ordinary simulation for alternating transition systems.

2 Basic Definitions

We fix a finite, non-empty set of propositions Σ with typical elements $a, b, c, \ldots$. *LTL formulas over* Σ are defined inductively by (1) tt and a are LTL formulas for every $a \in \Sigma$, and (2) if ϕ and ψ are LTL formulas, then so are $\neg\phi$, $\phi \vee \psi$, $\mathbf{X}\phi$ and $\phi\mathbf{U}\psi$.

LTL formulas are interpreted over ω-words, i. e., over infinite sequences of subsets of Σ. For every ω-word $w : \omega \to 2^\Sigma$, we define the relation $\models$ as follows.

$$w \models \quad \mathrm{tt}, \tag{1}$$
$$w \models \quad a \quad \text{iff } a \in w(0), \tag{2}$$
$$w \models \quad \neg\phi \quad \text{iff } w \not\models \phi, \tag{3}$$
$$w \models \phi \vee \psi \text{ iff } w \models \phi \text{ or } w \models \psi, \tag{4}$$
$$w \models \quad \mathbf{X}\phi \quad \text{iff } w[1..] \models \phi, \tag{5}$$
$$w \models \phi \mathbf{U} \psi \text{ iff } \exists i (w[i..] \models \psi \ \wedge \ \forall j < i (w[j..] \models \phi)), \tag{6}$$

where $w[i..]$ is defined by $w[i..](n) = w(i+n)$ for every $n < \omega$. As usual, we will allow derived logical operators like ff, $\wedge$, $\rightarrow$, and the temporal operators $\mathbf{V}$, $\mathbf{F}$, $\mathbf{G}$ defined by $\phi \mathbf{V} \psi = \neg(\neg\phi \mathbf{U} \neg\psi)$, $\mathbf{F}\phi = \mathrm{tt}\mathbf{U}\phi$ and $\mathbf{G}\phi = \mathrm{ff}\mathbf{V}\phi$. The language of an LTL formula ϕ is

$$L(\phi) = \{w \in (2^\Sigma)^\omega \mid w \models \phi\} \ . \tag{7}$$

Following [EH00], the transitions of our automata are labeled by so-called *terms* over the set of propositions. A term is the (possibly empty) conjunction of positive and negative propositions, i. e., the set of terms over Σ is

$$\mathrm{term}_\Sigma := \{ \bigwedge_{p \in M} p \wedge \bigwedge_{q \in N} \neg q \mid M, N \subseteq \Sigma\} \ . \tag{8}$$

For every $\Gamma \subseteq \Sigma$, let $t(\Gamma) = \bigwedge_{p \in \Gamma} p \wedge \bigwedge_{q \in \Sigma - \Gamma} \neg q$. We define a preorder $\sqsubseteq$ on term_Σ by setting $t_0 \sqsubseteq t_1$ iff $t_0 \rightarrow t_1$ is a tautology (for every $t_0, t_1 \in \mathrm{term}_\Sigma$).

A *nondeterministic Büchi automaton* (NBA for short) over Σ is a tuple

$$A \ = (Q, \Sigma, q_I, \Delta, F) \tag{9}$$

where Q is a finite set of states, $q_I \in Q$ is an initial state, $\Delta \subseteq Q \times \mathrm{term}_\Sigma \times Q$ a transition relation, and $F \subseteq Q$ a set of accepting states. Such an automaton A accepts a word $w : \omega \to 2^\Sigma$ if and only if there is a sequence of states $(q_i)_{i<\omega}$ of A such that $q_0 = q_I$ and for every $i < \omega$, there is a t_i such that $(q_i, t_i, q_{i+1}) \in \Delta$ and $t(w(i)) \sqsubseteq t_i$, and there are infinitely many $i < \omega$ such that $q_i \in F$. The *language of* A is

$$L(A) = \{w \in (2^\Sigma)^\omega \mid A \text{ accepts } w\} \ . \tag{10}$$

For $q \in Q$, we will write $A(q)$ for the automaton A with new initial state q, i. e., $A(q) = (Q, \Sigma, q, \Delta, F)$.

3 From LTL to Alternating Büchi Automata with ε-Transitions

In the first step of our algorithm, after a straightforward conversion to negation normal form, the LTL formula is translated to an *alternating Büchi automaton with term- and ε-transitions*, or ε-ABA for short. An ε-ABA is similar to an NBA as defined in Section 2, but with the following changes.

- There is a partition of the set of states Q into the sets E and U, called the *existential* and *universal* states respectively.
- The transition relation Δ is a subset of $Q \times (\mathrm{term}_\Sigma \cup \{\varepsilon\}) \times Q$, that is, we also allow transitions labeled with the empty word.

We say that an ε-ABA A is *legal* if there is no non-empty sequence $(q_i)_{i<n}$ of states of A such that $(q_i, \varepsilon, q_{i+1}) \in \Delta$ for every $i < n-1$ and $q_0 = q_{n-1}$. Throughout this paper, we will only consider legal ε-ABA.

We define the language of a legal ε-ABA $A = (Q, \Sigma, q_I, \Delta, E, U, F)$ as follows, using a game-theoretic approach. Given an ω-word w, the *word game* $G(A,)w$ is the Büchi game

$$(P, P_0, P_1, p_I, Z, F') \tag{11}$$

where $P = Q \times \omega \times \{0,1\}$ is the set of positions, $P_0 = U \times \omega \times \{0,1\}$ is the set of positions of Player 0 (called *Pathfinder*), $P_1 = E \times \omega \times \{0,1\}$ is the set of positions of Player 1 (called *Automaton*), $p_I = (q_I, 0, 1)$ is the initial position, $Z = \{((s,i,j),(s',i,0)) \mid (s,\varepsilon,s') \in \Delta\} \cup \{((s,i,j),(s',i+1,1)) \mid \exists t' \in \mathrm{term}_\Sigma : t(w(i)) \sqsubseteq t' \wedge (s,t',s') \in \Delta\}$ is the set of moves, and $F' = F \times \omega \times \{1\}$ is the set of accepting positions.

That is, the word game can be viewed as being played in rounds, and a round ends if one of the players chooses a transition labeled by a term, in which case the third component of a position switches from 0 to 1 (in a legal ε-ABA, every round is finite). The winner is determined by the sequence of the initial states of the rounds: The main difference from other definitions is that not all visited states are taken into account for acceptance. The language $L(A)$ of an ε-ABA A is

$$L(A) = \{w \in (2^\Sigma)^\omega \mid \text{Automaton wins } G(A,w)\} \ . \tag{12}$$

The translation from LTL to ε-ABA is straightforward using the well-known equivalences $\phi \mathbf{U} \psi \equiv \psi \vee (\phi \wedge \mathbf{X}(\phi \mathbf{U} \psi))$ and, dually, $\phi \mathbf{V} \psi \equiv \psi \wedge (\phi \vee \mathbf{X}(\phi \mathbf{V} \psi))$, see, e. g., [Var96]. That is, for an LTL formula ϕ_0, we define the ε-ABA $A(\phi_0)$ inductively as follows.

1. The initial state is ϕ_0.
2. If ϕ is a state of $A(\phi_0)$ then
   - if $\phi = \mathrm{tt}$, $(\mathrm{tt}, \mathrm{tt}, \mathrm{tt}) \in \Delta$ is a transition of $A(\phi_0)$,
   - if $\phi = a$ or $\phi = \neg a$ for $a \in \Sigma$, then $(\phi, \phi, \mathrm{tt}) \in \Delta$ and $\mathrm{tt} \in Q$,
   - if $\phi = \psi \vee \rho$ or $\phi = \psi \wedge \rho$, then $(\phi, \varepsilon, \psi), (\phi, \varepsilon, \rho) \in \Delta$ and $\psi, \rho \in Q$,
   - if $\phi = \mathbf{X}\psi$, then $(\phi, \mathrm{tt}, \psi) \in \Delta$ and $\psi \in Q$,
   - if $\phi = \psi \mathbf{U} \rho$, then $(\phi, \varepsilon, \rho), (\phi, \varepsilon, \psi \wedge \mathbf{X}\phi) \in \Delta$ and $\rho, \psi \wedge \mathbf{X}\phi \in Q$,
   - if $\phi = \psi \mathbf{V} \rho$, then $(\phi, \varepsilon, \rho), (\phi, \varepsilon, \psi \vee \mathbf{X}\phi) \in \Delta$ and $\rho, \psi \vee \mathbf{X}\phi \in Q$,
   - if $\phi = \mathbf{F}\psi$ or $\phi = \mathbf{G}\psi$, then $(\phi, \mathrm{tt}, \phi), (\phi, \varepsilon, \psi) \in \Delta$ and $\psi \in Q$.

Formulas of the form $\psi \wedge \rho$, $\psi \mathbf{V} \rho$ and $\mathbf{G}\psi$ are universal states, all other formulas are existential states. The set of accepting states contains all states of the form tt, $\psi \mathbf{V} \rho$, $\mathbf{G}\psi$. We also add states of the form $\psi \vee \rho$ and $\mathbf{X}(\psi \mathbf{V} \rho)$ to the accepting states. This does not change the language of the automaton, but with this definition, $A(\phi_0)$ is a *weak* alternating Büchi automaton [MSS86]. This property allows us to faster solve the simulation game (see Section 4).

Proposition 1 (cf. [Var96]). *For all LTL formulas ϕ over Σ, $L(\phi) = L(A(\phi))$.*

In our implementation, we also apply some simple rules to eliminate some of the ε-transitions in the computed ε-ABA. We will not further discuss this here.

4 The Simulation Game for ε-ABA

Next, we construct a simulation relation on the states of legal ε-ABA. This relation will be our main tool for on-the-fly simplifications.

Following [FW02b], our definition of simulation relations for ε-ABA is based on a basic game for which different winning conditions can be defined.
Let $A = (Q, \Sigma, q_I, \Delta, E, U, F)$ be a legal ε-ABA and $p_0, q_0 \in Q$. The *basic simulation game* $G(p_0, q_0)$ is played in rounds by two players, called *Spoiler* and *Duplicator*, who move two pebbles (a red and a green pebble) on the transition graph of A. A round ends if a term-labeled transition has been chosen for both pebbles; before that, arbitrarily many ε-labeled transitions can be chosen. We say that a pebble is *free* if no term-labeled transition has been chosen for it in the current round; else it is *locked*. At the beginning of a round, let the red pebble be placed on p and the green pebble on q. Then, the players play as follows.

1. Spoiler chooses a term $t \in \mathrm{term}_\Sigma$.
2. The progression of the round depends on the modes of p and q, and on the statuses of the pebbles (free or locked). Initially, both pebbles are free, and the round ends if both pebbles are locked. A player moves a free pebble on a state s by choosing a transition $(s, x, s') \in \Delta$ such that $x = \varepsilon$ or $x \in \mathrm{term}_\Sigma$ and $t \sqsubseteq x$. The round continues with the pebble on s'. If $x \in \mathrm{term}_\Sigma$, the moved pebble becomes locked for the remainder of the round. The following rules determine who of the players has to move which pebble.
   - If $p \in E$ and $q \in U$ and both pebbles are free, Spoiler moves one of the pebbles (he can choose which one).
   - If $p \in E$ and the red pebble is free, but $q \in E$ or the green pebble is locked, Spoiler has to move the red pebble.
   - Conversely, if $p \in U$ or the red pebble is locked, but $q \in U$ and the green pebble is free, Spoiler has to move the green pebble.

   If these cases do not apply, Duplicator has to move a free pebble in a symmetric fashion, as follows.
   - If $p \in U$ and $q \in E$ and both pebbles are free, Duplicator chooses one of the pebbles and moves it.
   - If $p \in U$ and the red pebble is free while the green pebble is locked, Duplicator moves the red pebble.
   - And if $q \in E$, the green pebble is free, and the red pebble is locked, Duplicator moves the green pebble.

If a player has to move a pebble but cannot (because there is no appropriate transition), he loses early. Or else the sequence $(p_i, q_i)_{i<\omega}$ of the initial positions of the rounds determines the winner (cf. the rules of the word game in Section 3).

We use the winning condition for *delayed* simulation, i. e., Duplicator wins if, for every $n < \omega$, if $p_n \in F$ then there is an $m \geq n$ such that $q_m \in F$. The simulation game with this winning condition is denoted $G^{de}(p_0, q_0)$.

Definition 1. *The relation* $\leq_{de} \subseteq Q \times Q$ *is defined by*

$$p \leq_{de} q \qquad \textit{iff} \qquad \textit{Duplicator wins } G^{de}(p, q) \ . \tag{13}$$

The relation $\leq_{de}$ is a preorder, i. e., it is reflexive and transitive. For every legal ε-ABA A and for all states p, q of A, if $p \leq_{de} q$ then $L(A(p)) \subseteq L(A(q))$ (cf. [FW02b]).

As a preorder, $\leq_{de}$ induces an equivalence relation $\equiv_{de}$ defined by

$$p \equiv_{de} q \qquad \text{iff} \qquad p \leq_{de} q \text{ and } q \leq_{de} p \ . \tag{14}$$

The following observations are important for an efficient computation of the simulation relation and for its use in an on-the-fly fashion.

Note that ε-ABA constructed according to Sect. 3 have a special structure. No strongly connected component of such an ε-ABA contains two equivalent states:

Proposition 2. *Let* ϕ *be an LTL formula, and let* C *be a strongly connected component of the transition graph of* $A(\phi)$*. Let* $q_0, q_1 \in C$*. If* $q_0 \equiv_{de} q_1$ *then* $q_0 = q_1$*.*

We will use this property in Section 5 for our on-the-fly simplifications.

Since the ε-ABA that we construct for LTL formulas are weak (see Sect. 3), our delayed simulation game is in fact a reachability game in an AND/OR-graph and can thus be solved asymptotically as fast as a direct simulation game, see [FW02b, Theorem 9].

If Spoiler can choose an arbitrary term at the beginning of a round, the size of the game graph is exponential in $|\Sigma|$, but this is not necessary. To reduce the size of the game, in our implementation the set of terms S that Spoiler can choose from depends on the current position ϕ of the red pebble, i. e., $S = S(\phi)$. For example, if the red pebble is on a state $\mathbf{X}\psi$ then $S(\mathbf{X}\psi) = \{\mathrm{tt}\}$, and $S(\psi \mathbf{U} \rho) = S(\psi) \cup S(\rho)$. Nevertheless, we cannot completely avoid an exponential blow-up here, because

$$S(\psi \mathbf{V} \rho) = S(\rho) \cup \{\psi' \wedge \rho' \mid \psi' \in S(\psi), \rho' \in S(\rho)\} \ . \tag{15}$$

That is, for an LTL formula ϕ of length n the ε-ABA $A(\phi)$ has $O(n)$ states and $O(n^2)$ transitions.

But the simulation game graph of $G^{de}(\phi, \phi)$ has $2^{O(k \log n)}$ positions and moves, where k is the maximal nesting depth of the operator $\mathbf{V}$ (e. g., for $(a \mathbf{V} b) \mathbf{V} (c \mathbf{V} (a \wedge (b \mathbf{V} c)))$, k is 3).

5 Computing the Nondeterministic Büchi Automaton with On-the-Fly Simplifications

We use an algorithm of Miyano and Hayashi [MH84] for translating an alternating to a nondeterministic Büchi automaton. While running this algorithm, we apply a set of simplification rules on-the-fly, i. e., after the construction of every single state of the NBA.

We first recall the classical construction of Miyano and Hayashi and then give some necessary definitions before presenting our algorithm.

The Miyano-Hayashi construction eliminates universal branchings via a modified power set construction. Given an ABA $A = (Q, \Sigma, q_I, \Delta, E, U, F)$, a state of the resulting NBA $A_{nba} = (Q', \Sigma, q'_I, \Delta', F')$ is of the form (M, N) with $N \subseteq M \subseteq Q$ and $N \cap F = \emptyset$. Informally, if the computation of the A_{nba} is in a state (M, N), the computation of A is in all the states of M simultaneously, and A_{nba} will be in an accepting state as soon as the computation branches ending in the states of $N \subseteq M$ reach accepting states (especially, (M, N) is an accepting state if $N = \emptyset$). For more details on the Miyano-Hayashi construction, see [MH84].

It is not necessary to find a definition for quotients of ε-ABA, because every possible minimization effect of such a quotienting is taken into account by our modified Miyano-Hayashi construction.

Instead of merging equivalent states, we will choose a standard representative for every simulation equivalence class. This representative is chosen topologically maximal in the transition graph of A, i. e., if q' is our representative for an equivalence class $[q]^{de}_{\equiv}$ then no state in $[q]^{de}_{\equiv} \setminus \{q'\}$ is reachable from q'. By Proposition 2, there is such a representative. The strongly connected components of the transition graph of A and their topological sorting can be computed in time $O(|Q| \cdot |\Delta|)$ (or $O(|\phi|^3)$, for an LTL formula ϕ and $A = A(\phi)$). Let $r \colon Q \to Q$ be the function that maps every state to the representative of its class.

We define the preorders $\leq'_{de}$ and $\leq_{\text{weak}}$ on Q' as follows. The relation $\leq'_{de} \subseteq Q' \times Q'$ is defined by

$$(M_0, N_0) \leq'_{de} (M_1, N_1) \text{ iff } \forall q \in M_1 \exists p \in M_0 : p \leq_{de} q \ \wedge \ (q \in N_1 \to p \in N_0) \ . \tag{16}$$

The relation $\leq_{\text{weak}} \subseteq Q' \times Q'$ is defined by

$$(M_0, N_0) \leq_{\text{weak}} (M_1, N_1) \qquad \text{iff} \qquad M_1 \subseteq M_0 \text{ and } N_1 \subseteq N_0 \ . \tag{17}$$

Note that $\leq_{\text{weak}}$ is a subset of $\leq'_{de}$, and $\leq'_{de}$ in turn is a subset of the full delayed simulation relation $\leq_{de}$ on Q' (which we do not compute).

The translation algorithm from A to A_{nba}

1. The initial state q'_I of A_{nba} is $(\{r(q_I)\}, \{r(q_I)\})$ or, if $r(q_I) \in F$, $(\{r(q_I)\}, \emptyset)$. We add q'_I to the auxiliary set of new states, i. e., $newStates := \{q'_I\}$.
2. Choose a state $(M, N) \in newStates$, remove it from $newStates$ and add it to Q'.

3. Compute the transitions starting at (M, N) using the method of Miyano and Hayashi. Let $T_{(M,N)} \subseteq \text{term}_\Sigma \times (2^Q \times 2^Q)$ be the set of these transitions.
4. Replace every transition $(t, (K, L)) \in T_{(M,N)}$ with $(t, (K', L'))$ where $K' = \min^{de}(r(K)) \cup \min^{de}(r(L))$ and $L' = \min^{de}(r(L)) - F$, where $\min^{de}(R) = \{q \in R \mid \forall r \in R : r \leq_{de} q \rightarrow r \equiv_{de} q\}$ for every $R \subseteq Q$. This is justified by [FW02a, Cor. 8].
5. Delete every transition $(t, (K, L))$ from $T_{(M,N)}$ for which there is another transition $(t', (K', L')) \in T_{(M,N)}$ such that $t \sqsubseteq t'$, and either $(K, L) \leq_{\text{weak}} (K', L')$ or $(K, L) \leq'_{de} (K', L')$ and $N = \emptyset$ or $L' = \emptyset$ (see [FW02a, Subsect. 7.4]).
6. For all $(t, (K, L)) \in T_{(M,N)}$, add $((M, N), t, (K, L))$ to Δ', and add (K, L) to *newStates* if $(K, L) \notin Q'$.
7. If $newStates \neq \emptyset$, continue with step 2.
8. The set of accepting states is $F' = \{(M, N) \in Q' \mid N = \emptyset\}$.

The correctness of the above algorithm follows from Proposition 1 together with [MH84, FW02a].

Theorem 1. *For every LTL formula ϕ, $L(\phi) = L(A_{nba}(\phi))$.*

In our implementation, we also apply some standard simplifications to the nondeterministic Büchi automaton which are not based on simulation but mainly on the structure of its strongly connected components. We will not elaborate on these techniques here. See [GO01] for more details.

6 Experimental Results

Our prototypical implementation "LTL → NBA" of the algorithm outlined in Sections 3 to 5 is accessible via CGI, see [FT]. See Figure 1 for a graphical sample output of LTL → NBA using the dot program of the Graphviz package [ATT].

We have used the very helpful tool *lbtt* of Tauriainen and Heljanko [TH02] to compare our implementation with the programs "Temporal Massage Parlor" of Etessami [Ete] and LTL2BA of Gastin and Oddoux [Odd]. With LTL2BA, we also performed tests with formula rewriting disabled.

The tool *lbtt* generates random LTL formulas. The user can adjust the length of the formulas and the relative frequency of the temporal operators.

The computation times should be read as rough indicators since our prototype is written in Python, which is a rather slow interpreted language, i. e., we suppose that an equivalent C program would be at least 10 times faster. The program TMP is written in ML while LTL2BA is implemented in C.

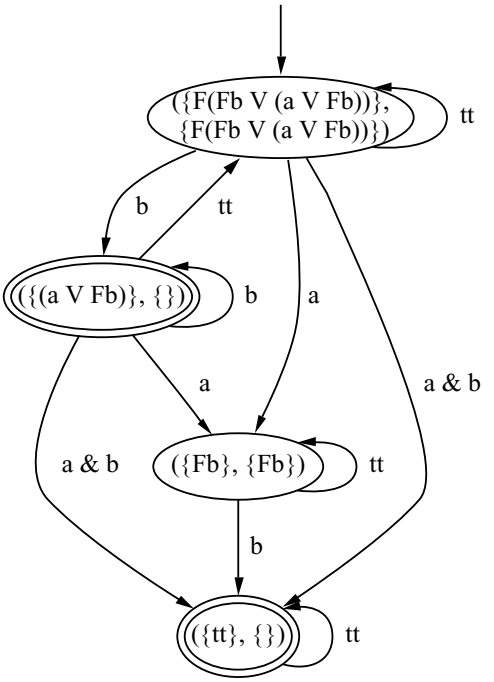

Fig. 1. Graphical output of LTL → NBA for the formula $\mathbf{F}((\mathbf{F}b)\mathbf{V}(a\mathbf{V}(\mathbf{F}b)))$

Test 1. 1000 formulas and their negations, of length 8 to 10, with at most 3 different propositions and an equal frequency of the operators ∨, ∧, **F**, **G**, **U**, **V**.

Test 1	LTL → NBA	TMP	LTL2BA	LTL2BA without f. r.
Avg. no. of states	3.54	3.59	3.76	4.61
Avg. no. of transitions	6.96	6.65	7.75	11.03
Total time (sec)	209.1	70.2	12.7	12.5

Test 2. 1000 formulas and their negations, now of length 10 to 14.

Test 2	LTL → NBA	TMP	LTL2BA	LTL2BA without f. r.
Avg. no. of states	4.37	4.71	4.93	6.58
Avg. no. of transitions	10.07	9.71	12.28	19.86
Total time (sec)	425.0	197.1	12.9	13.4

Test 3. Again 1000 formulas and their negations, now of length 11 to 15 and with an increased frequency of the operators **U** and **V**.

Test 3	LTL → NBA	TMP	LTL2BA	LTL2BA without f. r.
Avg. no. of states	5.06	5.71	5.74	8.18
Avg. no. of transitions	12.59	13.83	15.98	28.86
Total time (sec)	575.0	14737.1[1]	13.1	12.9

Test 4. 1000 formulas and their negations, of length 15, with the same frequency of the operators **U** and **V** as in Test 3.

Test 4	LTL → NBA	TMP	LTL2BA	LTL2BA without f. r.
Avg. no. of states	5.80	6.55	6.68	10.29
Avg. no. of transitions	16.05	16.86	20.63	42.93
Total time (sec)	986.6	3001.9	13.4	26.0

Our tests demonstrate that our implementation yields a competitive quality of results. With the complexity of the formulas increasing, our implementation becomes faster than TMP. LTL2BA is extremely fast, but the resulting automata are considerably larger. The differences in the performance of LTL2BA with and without formula rewriting may indicate the potential of adding this feature to our implementation, although the effect will probably be less dramatic since many rewrite rules are implicitly contained in our approach.

It may be surprising that our automata are often smaller than the automata computed by TMP although TMP computes the full delayed simulation quotient. We suppose this is because the algorithm of [GPVW95] used in TMP produces automata that are less suited for simulation-based minimizations than the automata we compute via alternating automata and the Miyano-Hayashi construction.

For detailed testing protocols, see [Fri].

7 Conclusion

We have presented a new algorithm for the translation of LTL formulas to nondeterministic Büchi automata which makes use of simulation relations for alternating Büchi automata. Our prototypical implementation demonstrates the power of these simulation relations as a tool for state-space reduction: The computed automata are small by comparison (although we do not use formula rewriting), and the average computation is fast in comparison to a simulation-based approach as used in TMP.

The simulation relations can be used on-the-fly and can easily be combined with other simplification techniques not based on simulation, like formula rewriting and SCC-based simplifications.

[1] TMP spent 13997.3 sec (nearly 4 hrs.) on the formula $\neg(((\mathbf{X}b)\mathbf{U}(((\mathbf{G}a)\mathbf{V}c)\mathbf{U}(\mathbf{G}\neg a)))\mathbf{V}a)$, resulting in an automaton of 115 states (LTL → NBA: 7.5 sec, 50 states; LTL2BA: 0.04 sec, 83 states).

Acknowledgments

Thanks to Björn Teegen who implemented LTL → NBA's parser for the LTL formulas, the web interface, and the interface to the dot program.

References

[AHKV98] R. Alur, Th.A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In D. Sangiorgi and R. de Simone, eds., *9th Int. Conf. on Concurrency Theory (CONCUR '98)*, vol. 1466 of *LNCS*, pp. 163–178, 1998. 37

[AS03] A. Arnold and L. Santocanale. Ambigious classes in the games μ-calculus hierarchy. In A. D. Gordon, ed., *6th Int. Conf. on Foundations of Software Science and Computational Structures (FOSSACS '03)*, vol. 2620 of *LNCS*, pp. 70–86, 2003. 37

[ATT] AT & T Labs-Research. Graphviz. `http://www.research.att.com/sw/tools/graphviz/`. 43

[Blo] R. Bloem. Wring: an LTL to Buechi translator. URL: `http://vlsi.colorado.edu/~rbloem/wring.html`. 37

[DHW91] D. L. Dill, A. J. Hu, and H. Wong-Toi. Checking for language inclusion using simulation preorders. In Kim Guldstrand Larsen and Arne Skou, eds., *Computer Aided Verification, 3rd Int. Workshop, CAV '91*, vol. 575 of *LNCS*, pp. 255–265, 1991. 36

[EH00] K. Etessami and G. Holzmann. Optimizing Büchi automata. In C. Palamidessi, ed., *11th Int. Conf. on Concurrency Theory (CONCUR 2000)*, vol. 1877 of *LNCS*, pp. 153–167, 2000. 35, 37, 38

[ESW01] K. Etessami, R. Schuller, and Th. Wilke. Fair simulation relations, parity games, and state space reduction for Büchi automata. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, eds., *ICALP*, volume 2076 of *LNCS*, pp. 694–707. Springer-Verlag, 2001. 36, 37

[Ete] K. Etessami. Temporal massage parlor. `http://www1.bell-labs.com/project/TMP/`. 36, 37, 43

[Fri] C. Fritz. Home page. `http://www.ti.informatik.uni-kiel.de/~fritz/`. 45

[FT] C. Fritz and B. Teegen. LTL → BA. `http://www.ti.informatik.uni-kiel.de/~teegen/ABA-Simulation/ltl.cgi`. 36, 43

[FW02a] C. Fritz and Th. Wilke. Simulation relations for alternating Büchi automata. Tech. Rep. 2019, Institut für Informatik, Kiel University, July 2002. Extended version. Available at `http://www.ti.informatik.uni-kiel.de/~fritz/TechRep2019ext.ps`. 43

[FW02b] C. Fritz and Th. Wilke. State space reductions for alternating Büchi automata: Quotienting by simulation equivalences. In M. Agrawal and A. Seth, eds., *22nd Conf. on Foundations of Software Technology and Theoretical Computer Science*, vol. 2556 of *LNCS*, pp. 157–168, 2002. 36, 37, 40, 41

[GBS02] S. Gurumurthy, R. Bloem, and F. Somenzi. Fair simulation minimization. In E. Brinksma and K. Guldstrand Larsen, eds., *Computer Aided Verification. 14th International Conference, CAV 2002*, vol. 2404 of *LNCS*, pp. 610–623, 2002. 36, 37

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979. 35

[GO01] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In G. Berry, H. Comon, and A. Finkel, eds., *Computer Aided Verification. 13th International Conference, CAV 2001*, vol. 2102 of *LNCS*, pp. 53–65. Springer-Verlag, 2001. 37, 43

[GPVW95] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. of the 15th Workshop on Protocol Specification, Testing, and Verification*, pp. 3–18, Warsaw, Poland, June 1995. Chapman Hall. 37, 45

[HHK95] M. Henzinger Rauch, Th.A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *36th Ann. Symp. on Foundations of Computer Science (FOCS '95)*, pp. 453–462, 1995. 37

[HKR97] Th.A. Henzinger, O. Kupferman, and S. K. Rajamani. Fair simulation. In *CONCUR '97*, vol. 1243 of *LNCS*, pp. 273–287, 1997. 37

[Hol] G. J. Holzmann. The SPIN homepage. `http://netlib.bell-labs.com/netlib/spin/whatispin.html`. 37

[HR00] Th.A. Henzinger and S. K. Rajamani. Fair bisimulation. In S. Graf and M. Schwartzbach, eds., *TACAS '00: Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1785 of *LNCS*, pp. 299–314. Springer-Verlag, 2000. 37

[Jur00] M. Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, eds., *STACS 2000, 17th Ann. Symp. on Theoretical Aspects of Computer Science*, vol. 1770 of *LNCS*, pp. 290–301, 2000. Springer-Verlag. 37

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984. 36, 42, 43

[Mil71] R. Milner. An algebraic definition of simulation between programs. In D. C. Cooper, editor, *Proc. of the 2nd Int. Joint Conf. on Artificial Intelligence*, pp. 481–489, London, UK, September 1971. William Kaufmann. ISBN 0-934613-34-6. 37

[Mil80] R. Milner. *A Calculus of Communicating Systems*, vol. 92 of *LNCS*. Springer-Verlag, 1980. 37

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992. 35

[MSS86] D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th Int. Coll. on Automata, Languages, and Programming (ICALP '86)*, vol. 226 of *LNCS*, 1986. 39

[Odd] D. Oddoux. LTL2BA. `http://verif.liafa.jussieu.fr/cgi-bin/binbin/ltl2ba.shtml`. 36, 37, 43

[Par81] D. M. R. Park. Concurrency and automata on infinite sequences. In P. Deussen, ed., *5th GI Conference*, vol. 104 of *LNCS*, pp. 167–183, 1981. 37

[SB00] F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In E. Allen Emerson and A. Prasad Sistla, eds., *Computer Aided Verification, 12th International Conference (CAV 2000)*, vol. 1855 of *LNCS*, pp. 248–263, 2000. 37

[TH02] H. Tauriainen and K. Heljanko. Testing LTL formula translation into Büchi automata. *Int. Journal on Software Tools for Technology Transfer*, 4(1):57–70, 2002. 36, 43

[Var96] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, vol. 1043 of *LNCS*, pp. 238–266, 1996. 39, 40