Effective Translation of LTL to Deterministic Rabin Automata: Beyond the (F,G)-Fragment*

Tomáš Babiak, František Blahoudek, Mojmír Křetínský, and Jan Strejček

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xbabiak, xblahoud, kretinsky, strejcek}@fi.muni.cz

Abstract. Some applications of linear temporal logic (LTL) require to translate formulae of the logic to deterministic ω-automata. There are currently two translators producing deterministic automata: `ltl2dstar` working for the whole LTL and Rabinizer applicable to LTL(F, G) which is the LTL fragment using only modalities F and G. We present a new translation to deterministic Rabin automata via alternating automata and deterministic transition-based generalized Rabin automata. Our translation applies to a fragment that is strictly larger than LTL(F, G). Experimental results show that our algorithm can produce significantly smaller automata compared to Rabinizer and `ltl2dstar`, especially for more complex LTL formulae.

1 Introduction

Linear temporal logic (LTL) is a popular formalism for specification of behavioral system properties with major applications in the area of model checking [8, 5]. More precisely, LTL is typically used as a human-oriented front-end formalism as LTL formulae are succinct and easy to write and understand. Model checking algorithms usually work with an ω-automaton representing all behaviors violating a given specification formula rather than with the LTL formula directly. Hence, specifications written in the form of LTL formulae are negated and translated to equivalent ω-automata [31]. There has been a lot of attention devoted to translation of LTL to *nondeterministic Büchi automata (NBA)*, see for example [10, 11, 29, 15] and the research in this direction still continues [12, 4, 2]. However, there are algorithms that need specifications given by *deterministic* ω-automata, for example, those for LTL model checking of probabilistic systems [30, 9, 5] and those for synthesis of reactive modules for LTL specifications [7, 26], for a recent survey see [20]. As *deterministic Büchi automata (DBA)* cannot express all the properties expressible in LTL, one has to choose deterministic automata with different acceptance condition.

There are basically two approaches to translation of LTL to deterministic ω-automata. The first one translates LTL to NBA and then it employs Safra's construction [27] (or some of its variants or alternatives like [23, 28]) to transform the NBA into a deterministic automaton. This approach is represented by the

* The authors are supported by The Czech Science Foundation, grant P202/12/G061.

tool `ltl2dstar` [16] which uses an improved Safra's construction [17, 18] usually in connection with LTL to NBA translator LTL2BA [15]. The main advantage of this approach is its universality: as LTL2BA can translate any LTL formula into an NBA and the Safra's construction can transform any NBA to a *deterministic Rabin automaton (DRA)*, `ltl2dstar` works for the whole LTL. The main disadvantage is also connected with the universality: the determinization step does not employ the fact that the NBA represents only an LTL definable property. One can easily observe that `ltl2dstar` produces unnecessarily large automata, especially for formulae with more fairness subformulae.

The second approach is to avoid Safra's construction. As probabilistic model-checkers deal with linear arithmetic, they do not profit from symbolically represented deterministic automata of [24, 22]. A few translations of some simple LTL fragments to DBA have been suggested, for example [1]. Recently, a translation of a significantly larger LTL fragment to DRA has been introduced in [19] and subsequently implemented in the tool Rabinizer [14]. The algorithm builds a *generalized deterministic Rabin automata (GDRA)* directly from a formula. A DRA is then produced by a degeneralization procedure. Rabinizer often produces smaller automata than `ltl2dstar`. The main disadvantage is that it works for LTL(F, G) only, i.e. the LTL fragment containing only temporal operators *eventually* (F) and *always* (G). Authors of the translation claim that it can be extended to a fragment containing also the operator *next* (X).

In this paper, we present another Safraless translation of an LTL fragment to DRA. The translation is influenced by the successful LTL to NBA translation algorithm LTL2BA [15] and it proceeds in the following three steps:

1. A given LTL formula φ is translated into a *very weak alternating co-Büchi automaton (VWAA)* $\mathcal{A}$ as described in [15]. If φ is an LTL($\mathsf{F_s}, \mathsf{G_s}$) formula, i.e. any formula which makes use of F, G, and their strict variants $\mathsf{F_s}$ and $\mathsf{G_s}$ as the only temporal operators, then $\mathcal{A}$ satisfies an additional structural condition. We call such automata *may/must alternating automata (MMAA)*.
2. The MMAA $\mathcal{A}$ is translated into a *transition-based generalized deterministic Rabin automaton (TGDRA)* $\mathcal{G}$. The construction of generalized Rabin pairs of $\mathcal{G}$ is inspired by [19].
3. Finally, $\mathcal{G}$ is degeneralized into a (state-based) DRA $\mathcal{D}$.

In summary, our contributions are as follows. First, note that the fragment LTL($\mathsf{F_s}, \mathsf{G_s}$) is strictly more expressive than LTL(F, G). Moreover, it can be shown that our translation works for a fragment even larger than LTL($\mathsf{F_s}, \mathsf{G_s}$) but still smaller than the whole LTL. Second, the translation has a slightly better theoretical bound on the size of produced automata comparing to `ltl2dstar`, but the same bound as Rabinizer. Experimental results show that, for small formulae, our translation typically produces automata of a smaller or equal size as the other two translators. However, for parametrized formulae, it often produces automata that are significantly smaller. Third, we note that our TGDRA are much smaller than the (state-based) GDRA of [14]. We conjecture that algorithms for model checking of probabilistic system, e.g. those in PRISM [21], can be adapted to work with TGDRA as they are adapted to work with GDRA [6].

2 Preliminaries

This section recalls the notion of linear temporal logic (LTL) [25] and describes the ω-automata used in the following.

Linear Temporal Logic (LTL) The syntax of LTL is defined by

$$\varphi \ ::= \ tt \ | \ a \ | \ \neg\varphi \ | \ \varphi \vee \varphi \ | \ \varphi \wedge \varphi \ | \ \mathsf{X}\varphi \ | \ \varphi \, \mathsf{U} \, \varphi,$$

where tt stands for *true*, a ranges over a countable set AP of *atomic propositions*, X and U are temporal operators called *next* and *until*, respectively. An *alphabet* is a finite set $\Sigma = 2^{AP'}$, where AP' is a finite subset of AP. An ω-*word* (or simply a *word*) over Σ is an infinite sequence of letters $u = u_0u_1u_2\ldots \in \Sigma^\omega$. By $u_{i..}$ we denote the suffix $u_{i..} = u_iu_{i+1}\ldots$.

We inductively define when a word u *satisfies* a formula φ, written $u \models \varphi$, as follows.

$$\begin{array}{lll}
u \models tt & & \\
u \models a & \text{iff} & a \in u_0 \\
u \models \neg\varphi & \text{iff} & u \not\models \varphi \\
u \models \varphi_1 \vee \varphi_2 & \text{iff} & u \models \varphi_1 \text{ or } u \models \varphi_2 \\
u \models \varphi_1 \wedge \varphi_2 & \text{iff} & u \models \varphi_1 \text{ and } u \models \varphi_2 \\
u \models \mathsf{X}\varphi & \text{iff} & u_{1..} \models \varphi \\
u \models \varphi_1 \, \mathsf{U} \, \varphi_2 & \text{iff} & \exists i \geq 0 \, . \, (\, u_{i..} \models \varphi_2 \text{ and } \forall\, 0 \leq j < i \, . \, u_{j..} \models \varphi_1 \,)
\end{array}$$

Given an alphabet Σ, a formula φ defines the language $L^\Sigma(\varphi) = \{u \in \Sigma^\omega \mid u \models \varphi\}$. We write $L(\varphi)$ instead of $L^{2^{AP(\varphi)}}(\varphi)$, where $AP(\varphi)$ denotes the set of atomic propositions occurring in the formula φ.

We define derived unary temporal operators *eventually* (F), *always* (G), *strict eventually* ($\mathsf{F_s}$), and *strict always* ($\mathsf{G_s}$) by the following equivalences: $\mathsf{F}\varphi \equiv tt \, \mathsf{U} \, \varphi$, $\mathsf{G}\varphi \equiv \neg\mathsf{F}\neg\varphi$, $\mathsf{F_s}\varphi \equiv \mathsf{XF}\varphi$, and $\mathsf{G_s}\varphi \equiv \mathsf{XG}\varphi$.

LTL(F, G) denotes the LTL fragment consisting of formulae built with temporal operators F and G only. The fragment build with temporal operators $\mathsf{F_s}$, $\mathsf{G_s}$, F and G is denoted by LTL($\mathsf{F_s}, \mathsf{G_s}$) as $\mathsf{F}\varphi$ and $\mathsf{G}\varphi$ can be seen as abbreviations for $\varphi \vee \mathsf{F_s}\varphi$ and $\varphi \wedge \mathsf{G_s}\varphi$, respectively. Note that LTL($\mathsf{F_s}, \mathsf{G_s}$) is strictly more expressive than LTL(F, G) as formulae $\mathsf{F_s}a$ and $\mathsf{G_s}a$ cannot be equivalently expressed in LTL(F, G).

An LTL formula is in *positive normal form* if no operator occurs in the scope of any negation. Each LTL($\mathsf{F_s}, \mathsf{G_s}$) formula can be transformed to this form using De Morgan's laws for $\wedge$ and $\vee$ and the equivalences $\neg\mathsf{F_s}\psi \equiv \mathsf{G_s}\neg\psi$, $\neg\mathsf{G_s}\psi \equiv \mathsf{F_s}\neg\psi$, $\neg\mathsf{F}\psi \equiv \mathsf{G}\neg\psi$, and $\neg\mathsf{G}\psi \equiv \mathsf{F}\neg\psi$. We say that a formula is *temporal* if its topmost operator is neither conjunction, nor disjunction (note that a and $\neg a$ are also temporal formulae).

Deterministic Rabin Automata and Their Generalization A *semiautomaton* is a tuple $\mathcal{T} = (S, \Sigma, \delta, s_I)$, where S is a finite set of *states*, Σ is an alphabet, $s_I \in S$ is the *initial state*, and $\delta \subseteq S \times \Sigma \times S$ is a deterministic *transition relation*, i.e. for each state $s \in S$ and each $\alpha \in \Sigma$, there is at most one state s' such that $(s, \alpha, s') \in \delta$. A triple $(s, \alpha, s') \in \delta$ is called a *transition* from s to s' labelled by α, or an α-transition of s leading to s'. In illustrations, all transitions with the same source state and the same target state are usually depicted by a single edge labelled by a propositional formula ψ over AP representing the corresponding transition labels (e.g. given $\Sigma = 2^{\{a,b\}}$, the formula $\psi = a \vee b$ represents labels $\{a\}, \{a, b\}, \{b\}$).

A *run* of a semiautomaton $\mathcal{T}$ over a word $u = u_0u_1 \ldots \in \Sigma^\omega$ is an infinite sequence $\sigma = (s_0, u_0, s_1)(s_1, u_1, s_2) \ldots \in \delta^\omega$ of transitions such that $s_0 = s_I$. By $\mathit{Inf}_t(\sigma)$ (resp. $\mathit{Inf}_s(\sigma)$) we denote the set of transitions (resp. states) occurring infinitely often in σ. For each word $u \in \Sigma^\omega$, a semiautomaton has at most one run over u denoted by $\sigma(u)$.

A *deterministic Rabin automaton* (DRA) is a tuple $\mathcal{D} = (S, \Sigma, \delta, s_I, \mathcal{R})$, where (S, Σ, δ, s_I) is a semiautomaton and $\mathcal{R} \subseteq 2^S \times 2^S$ is a finite set of *Rabin pairs*. Runs of $\mathcal{D}$ are runs of the semiautomaton. A run σ *satisfies* a Rabin pair $(K, L) \in \mathcal{R}$ if $\mathit{Inf}_s(\sigma) \cap K = \emptyset$ and $\mathit{Inf}_s(\sigma) \cap L \neq \emptyset$. A run is *accepting* if it satisfies some Rabin pair of $\mathcal{R}$. The language of $\mathcal{D}$ is the set $L(\mathcal{D})$ of all words $u \in \Sigma^\omega$ such that $\sigma(u)$ is accepting.

A *transition-based generalized deterministic Rabin automaton* (TGDRA) is a tuple $\mathcal{G} = (S, \Sigma, \delta, s_I, \mathcal{GR})$, where (S, Σ, δ, s_I) is a semiautomaton and $\mathcal{GR} \subseteq 2^\delta \times 2^{2^\delta}$ is a finite set of *generalized Rabin pairs*. Runs of $\mathcal{G}$ are runs of the semiautomaton. A run σ *satisfies* a generalized Rabin pair $(K, \{L_j\}_{j \in J}) \in \mathcal{GR}$ if $\mathit{Inf}_t(\sigma) \cap K = \emptyset$ and, for each $j \in J$, $\mathit{Inf}_t(\sigma) \cap L_j \neq \emptyset$. A run is *accepting* if it satisfies some generalized Rabin pair of $\mathcal{GR}$. The language of $\mathcal{G}$ is the set $L(\mathcal{G})$ of all words $u \in \Sigma^\omega$ such that $\sigma(u)$ is accepting.

A generalization of DRA called *generalized deterministic Rabin automata* (GDRA) has been considered in [19, 14]. The accepting condition of GDRA is a boolean combination (in disjunctive normal form) of Rabin pairs. A run σ is accepting if σ satisfies this condition.

Very Weak Alternating Automata and Their Subclass A *very weak alternating co-Büchi automaton* (VWAA) $\mathcal{A}$ is a tuple $(S, \Sigma, \delta, I, F)$, where S is a finite set of *states*, subsets $c \subseteq S$ are called *configurations*, Σ is an *alphabet*, $\delta \subseteq S \times \Sigma \times 2^S$ is an *alternating transition relation*, $I \subseteq 2^S$ is a non-empty set of *initial configurations*, $F \subseteq S$ is a set of *co-Büchi accepting states*, and there exists a partial order on S such that, for every transition $(s, \alpha, c) \in \delta$, all the states of c are lower or equal to s.

A triple $(s, \alpha, c) \in \delta$ is called a transition from s to c labelled by α, or an α-transition of s. We say that s is the *source state* and c the *target configuration* of the transition. A transition is *looping* if the target configuration contains the source state, i.e. $s \in c$. A transition is called a *selfloop* if its target configuration contains the source state only, i.e. $c = \{s\}$.

Figure 1(a) shows a VWAA that accepts the language described by the formula $\mathsf{G}(\mathsf{F_s} a \wedge \mathsf{F_s} b) \vee \mathsf{G} b$. Transitions are depicted by branching edges. If a target

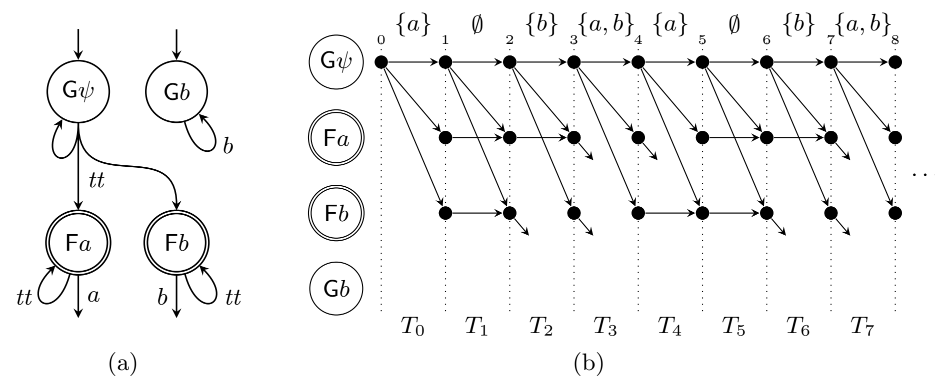

Fig. 1. (a) A VWAA (and also MMAA) corresponding to formula $\mathsf{G}\psi \vee \mathsf{G}b$, where $\psi = \mathsf{F_s}a \wedge \mathsf{F_s}b$. (b) An accepting run of the automaton over $(\{a\}\emptyset\{b\}\{a,b\})^\omega$.

configuration is empty, the corresponding edge leads to an empty space. We often depict all transitions with the same source state and the same target configuration by a single edge (as for semiautomata). Each initial configuration is represented by a possibly branching unlabelled edge leading from an empty space to the states of the configuration. Co-Büchi accepting states are double circled.

A *multitransition* T with a label α is a set of transitions with the same label and such that the source states of the transitions are pairwise different. A *source configuration* of T, denoted by $\mathrm{dom}(T)$, is the set of source states of transitions in T. A *target configuration* of T, denoted by $\mathrm{range}(T)$, is the union of target configurations of transitions in T. We define a *multitransition relation* $\Delta \subseteq 2^S \times \Sigma \times 2^S$ as

$$\Delta = \{(\mathrm{dom}(T), \alpha, \mathrm{range}(T)) \mid \text{there exists a multitransition } T \text{ with label } \alpha\}.$$

A *run* ρ of a VWAA $\mathcal{A}$ over a word $w = w_0w_1 \ldots \in \Sigma^\omega$ is an infinite sequence $\rho = T_0T_1 \ldots$ of multitransitions of $\mathcal{A}$ such that $\mathrm{dom}(T_0)$ is an initial configuration of $\mathcal{A}$ and, for each $i \geq 0$, T_i is labelled by w_i and $\mathrm{range}(T_i) = \mathrm{dom}(T_{i+1})$.

A run can be represented as a directed acyclic graph (DAG). For example, the DAG of Figure 1(b) represents a run of the VWAA of Figure 1(a). The dotted lines divide the DAG into segments corresponding to multitransitions. Each transition of a multitransition is represented by edges leading across the corresponding segment from the starting state to states of the target configuration. As our alternating automata are very weak, we can order the states in a way that all edges in any DAG go only to the same or a lower row.

An accepting run corresponds to a DAG where each branch contains only finitely many states from F. Formally, the run ρ is *accepting* if it has no suffix where, for some co-Büchi accepting state $f \in F$, each multitransition contains a looping transition from f. The language of $\mathcal{A}$ is the set $L(\mathcal{A}) = \{w \in \Sigma^\omega \mid \mathcal{A} \text{ has an accepting run of over } w\}$. By $\mathit{Inf}_s(\rho)$ we denote the set of states that occur in $\mathrm{dom}(T_i)$ for infinitely many indices i.

Definition 1. *A* may/must alternating automaton *(MMAA) is a VWAA where each state fits into one of the following three categories:*

1. May-states – *states with a selfloop for each $\alpha \in \Sigma$. A run that enters such a state* may *wait in the state for an arbitrary number of steps.*
2. Must-states – *every transition of a must-state is looping. A run that enters such a state can never leave it. In other words, the run* must *stay there.*
3. Loopless states – *states that have no looping transitions and no predecessors. They can appear only in initial configurations (or they are unreachable).*

The automaton of Figure 1(a) is an MMAA with must-states $\mathsf{G}\psi, \mathsf{G}b$ and may-states $\mathsf{F}a, \mathsf{F}b$.

We always assume that the set F of an MMAA coincides with the set of all may-states of the automaton. This assumption is justified by the following observations:

– There are no looping transitions of loopless states. Hence, removing all loopless states from F has no effect on acceptance of any run.
– All transitions leading from must-states are looping. Hence, if a run contains a must-state that is in F, then the run is non-accepting. Removing all must-states in F together with their adjacent transitions from an MMAA has no effect on its accepting runs.
– Every may-state has selfloops for all $\alpha \in \Sigma$. If such a state is not in F, we can always apply these selfloops without violating acceptance of any run. We can also remove these states from all the target configurations of all transitions of an MMAA without affecting its language.

3 Translation of LTL($\mathsf{F_s}, \mathsf{G_s}$) to MMAA

We present the standard translation of LTL to VWAA [15] restricted to the fragment LTL($\mathsf{F_s}, \mathsf{G_s}$). In this section, we treat the transition relation $\delta \subseteq S \times \Sigma \times 2^S$ of a VWAA as a function $\delta : S \times \Sigma \to 2^{2^S}$, where $c \in \delta(s, \alpha)$ means $(s, \alpha, c) \in \delta$. Further, we consider $\mathsf{G}\psi$ and $\mathsf{F}\psi$ to be subformulae of $\mathsf{G_s}\psi$ and $\mathsf{F_s}\psi$, respectively. This is justified by equivalences $\mathsf{G_s}\psi \equiv \mathsf{XG}\psi$ and $\mathsf{F_s}\psi \equiv \mathsf{XF}\psi$.

Let φ be an LTL($\mathsf{F_s}, \mathsf{G_s}$) formula in positive normal form. An equivalent VWAA is constructed as $\mathcal{A}_\varphi = (Q, \Sigma, \delta, I, F)$, where

– Q is the set of temporal subformulae of φ,
– $\Sigma = 2^{AP(\varphi)}$,
– δ is defined as

$$\begin{aligned} \delta(tt, \alpha) &= \{\emptyset\} & \delta(a, \alpha) &= \{\emptyset\} \text{ if } a \in \alpha, \emptyset \text{ otherwise} \\ \delta(\neg tt, \alpha) &= \emptyset & \delta(\neg a, \alpha) &= \{\emptyset\} \text{ if } a \notin \alpha, \emptyset \text{ otherwise} \\ \delta(\mathsf{G_s}\psi, \alpha) &= \{\{\mathsf{G}\psi\}\} & \delta(\mathsf{G}\psi, \alpha) &= \{c \cup \{\mathsf{G}\psi\} \mid c \in \overline{\delta}(\psi, \alpha)\} \\ \delta(\mathsf{F_s}\psi, \alpha) &= \{\{\mathsf{F}\psi\}\} & \delta(\mathsf{F}\psi, \alpha) &= \{\{\mathsf{F}\psi\}\} \cup \overline{\delta}(\psi, \alpha), \text{ where} \end{aligned}$$

$$\begin{aligned} \overline{\delta}(\psi, \alpha) &= \delta(\psi, \alpha) \text{ if } \psi \text{ is a temporal formula} \\ \overline{\delta}(\psi_1 \vee \psi_2, \alpha) &= \overline{\delta}(\psi_1, \alpha) \cup \overline{\delta}(\psi_2, \alpha) \\ \overline{\delta}(\psi_1 \wedge \psi_2, \alpha) &= \{c_1 \cup c_2 \mid c_1 \in \overline{\delta}(\psi_1, \alpha) \text{ and } c_2 \in \overline{\delta}(\psi_2, \alpha)\}, \end{aligned}$$

– $I = \overline{\varphi}$ where $\overline{\varphi}$ is defined as

$$\begin{aligned} \overline{\psi} &= \{\{\psi\}\} \text{ if } \psi \text{ is a temporal formula} \\ \overline{\psi_1 \vee \psi_2} &= \overline{\psi_1} \cup \overline{\psi_2} \\ \overline{\psi_1 \wedge \psi_2} &= \{O_1 \cup O_2 \mid O_1 \in \overline{\psi_1} \text{ and } O_2 \in \overline{\psi_2}\}, \text{ and} \end{aligned}$$

– $F \subseteq Q$ is the set of all subformulae of the form $\mathsf{F}\psi$ in Q.

Using the partial order "is a subformula of" on states, one can easily prove that $\mathcal{A}_\varphi$ is a VWAA. Moreover, all the states of the form $\mathsf{G}\psi$ are must-states and all the states of the form $\mathsf{F}\psi$ are may-states. States of other forms are loopless and they are unreachable unless they appear in I. Hence, the constructed automaton is also an MMAA. Figure 1(a) shows an MMAA produced by the translation of formula $\mathsf{G}(\mathsf{F_s}a \wedge \mathsf{F_s}b) \vee \mathsf{G}b$.

In fact, MMAA and LTL($\mathsf{F_s}, \mathsf{G_s}$) are expressively equivalent. The reverse translation can be found in the full version of this paper [3].

4 Translation of MMAA to TGDRA

In this section we present a translation of an MMAA $\mathcal{A} = (S, \Sigma, \delta_\mathcal{A}, I, F)$ with multitransition relation $\Delta_\mathcal{A}$ into an equivalent TGDRA $\mathcal{G}$. At first we build a semiautomaton $\mathcal{T}$ and then we describe the transition based generalized Rabin acceptance condition $\mathcal{GR}$ of $\mathcal{G}$.

4.1 Semiautomaton $\mathcal{T}$

The idea of our seminautomaton construction is straightforward: a run $\sigma(w)$ of the semiautomaton $\mathcal{T}$ tracks all runs of $\mathcal{A}$ over w. More precisely, the state of $\mathcal{T}$ reached after reading a finite input consists of all possible configurations in which $\mathcal{A}$ can be after reading the same input. Hence, states of the semiautomaton are sets of configurations of $\mathcal{A}$ and we call them *macrostates*. We use $f, s, s_1, s_2, \ldots$ to denote states of $\mathcal{A}$ (f stands for an accepting state of F), $c, c_1, c_2, \ldots$ to denote configurations of $\mathcal{A}$, and $m, m_1, m_2, \ldots$ to denote macrostates of $\mathcal{T}$. Further, we use $t, t_1, t_2 \ldots$ to denote the transitions of $\mathcal{A}$, $T, T_0, T_1 \ldots$ to denote multitransitions of $\mathcal{A}$, and $r, r_1, r_2 \ldots$ to denote the transitions of $\mathcal{T}$, which are called *macrotransitions* hereafter.

Formally, we define the *semiautomaton* $\mathcal{T} = (M, \Sigma, \delta_\mathcal{T}, m_I)$ for $\mathcal{A}$ as follows:

– $M \subseteq 2^{2^S}$ is the set *macrostates*, restricted to those reachable from the initial macrostate m_I by $\delta_\mathcal{T}$,
– $(m_1, \alpha, m_2) \in \delta_\mathcal{T}$ iff $m_2 = \bigcup_{c \in m_1} \{c' \mid (c, \alpha, c') \in \Delta_\mathcal{A}\}$, i.e. for each $m_1 \in M$ and $\alpha \in \Sigma$, there is a single macrotransition $(m_1, \alpha, m_2) \in \delta_\mathcal{T}$, where m_2 consists of target configurations of all α-multitransitions leading from configurations in m_1, and
– $m_I = I$ is the *initial macrostate*.

Figure 2 depicts the semiautomaton $\mathcal{T}$ for the MMAA of Figure 1(a). Each row in a macrostate represents one configuration.

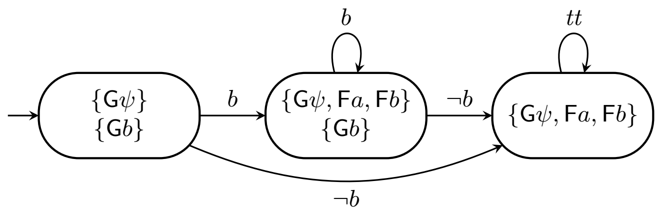

Fig. 2. The semiautomaton $\mathcal{T}$ for the MMAA of Figure 1(a).

4.2 Acceptance Condition $\mathcal{GR}$ of the TGDRA $\mathcal{G}$

For any subset $Z \subseteq S$, $must(Z)$ denotes the set of must-states of Z. An MMAA run ρ is *bounded by* $Z \subseteq S$ iff $\mathit{Inf}_s(\rho) \subseteq Z$ and $must(\mathit{Inf}_s(\rho)) = must(Z)$. For example, the run of Figure 1(b) is bounded by the set $\{\mathsf{G}\psi, \mathsf{F}a, \mathsf{F}b\}$.

For any fixed $Z \subseteq S$, we define the set $\mathrm{AC}_Z \subseteq 2^S$ of *allowed configurations* of $\mathcal{A}$ and the set $\mathrm{AT}_Z \subseteq \delta_{\mathcal{T}}$ of *allowed macrotransitions* of $\mathcal{T}$ as follows:

$$\mathrm{AC}_Z = \{c \subseteq Z \mid must(c) = must(Z)\}$$
$$\mathrm{AT}_Z = \{(m_1, \alpha, m_2) \in \delta_{\mathcal{T}} \mid \exists c_1 \in \mathrm{AC}_Z, c_2 \in (m_2 \cap \mathrm{AC}_Z) : (c_1, \alpha, c_2) \in \Delta_{\mathcal{A}}\}^1$$

Clearly, a run ρ of $\mathcal{A}$ is bounded by Z if and only if ρ has a suffix containing only configurations of AC_Z. Let ρ be a run over w with such a suffix. As the semiautomaton $\mathcal{T}$ tracks all runs of $\mathcal{A}$ over a given input, the run $\sigma(w)$ of $\mathcal{T}$ 'covers' also ρ. Hence, $\sigma(w)$ has a suffix where, for each macrotransition (m_i, w_i, m_{i+1}), there exist configurations $c_1 \in m_i \cap \mathrm{AC}_Z$ and $c_2 \in m_{i+1} \cap \mathrm{AC}_Z$ satisfying $(c_1, w_i, c_2) \in \Delta_{\mathcal{A}}$. In other words, $\sigma(w)$ has a suffix containing only macrotransitions of AT_Z. This observation is summarized by the following lemma.

Lemma 1. *If $\mathcal{A}$ has a run over w bounded by Z, then the run $\sigma(w)$ of $\mathcal{T}$ contains a suffix of macrotransitions of* AT_Z*.*

In fact, the other direction can be proved as well: if $\sigma(w)$ contains a suffix of macrotransitions of AT_Z, then $\mathcal{A}$ has a run over w bounded by Z.

For each $f \in F \cap Z$, we also define the set AT_Z^f as the set of all macrotransitions in AT_Z such that $\mathcal{A}$ contains a non-looping transition of f with the same label and with the target configuration not leaving Z:

$$\mathrm{AT}_Z^f = \{(m_1, \alpha, m_2) \in \mathrm{AT}_Z \mid \exists (f, \alpha, c) \in \delta_{\mathcal{A}} : f \notin c, c \subseteq Z\}$$

Using the sets AT_Z and AT_Z^f, we define one generalized Rabin pair $\mathcal{GR}_Z$ for each subset of states $Z \subseteq S$:

$$\mathcal{GR}_Z = (\delta_{\mathcal{T}} \smallsetminus \mathrm{AT}_Z, \{\mathrm{AT}_Z^f\}_{f \in F \cap Z}) \quad (1)$$

Lemma 2. *If there is an accepting run ρ of $\mathcal{A}$ over w then the run $\sigma(w)$ of $\mathcal{T}$ satisfies $\mathcal{GR}_Z$ for $Z = \mathit{Inf}_s(\rho)$.*

[1] A definition of AT_Z with $c_1 \in (m_1 \cap \mathrm{AC}_Z)$ would be more intuitive, but less effective.

Proof. As ρ is bounded by Z, Lemma 1 implies that $\sigma(w)$ has a suffix $r_i r_{i+1} \dots$ of macrotransitions of AT_Z. Thus $\mathit{Inf}_t(\sigma(w)) \cap (\delta_{\mathcal{T}} \smallsetminus \mathrm{AT}_Z) = \emptyset$.

As $Z = \mathit{Inf}_s(\rho)$ and $\rho = T_0 T_1 \dots$ is accepting, for each $f \in F \cap Z$, ρ includes infinitely many multitransitions T_j where $f \in \mathrm{dom}(T_j)$ and T_j contains a non-looping transition $(f, w_j, c) \in \delta_{\mathcal{A}}$ satisfying $f \notin c$ and $c \subseteq Z$. Hence, the corresponding macrotransitions r_j that are also in the mentioned suffix $r_i r_{i+1} \dots$ of $\sigma(w)$ are elements of AT_Z^f. Therefore, $\mathit{Inf}_t(\sigma(w)) \cap \mathrm{AT}_Z^f \neq \emptyset$ for each $f \in F \cap Z$ and $\sigma(w)$ satisfies $\mathcal{GR}_Z$. □

Lemma 3. *If a run $\sigma(w)$ of $\mathcal{T}$ satisfies $\mathcal{GR}_Z$ then there is an* accepting *run of $\mathcal{A}$ over w bounded by Z.*

Proof. Let $\sigma(w) = r_0 r_1 \dots$ be a run of $\mathcal{T}$ satisfying $\mathcal{GR}_Z$, i.e. $\sigma(w)$ has a suffix of macrotransitions of AT_Z and $\sigma(w)$ contains infinitely many macrotransitions of AT_Z^f for each $f \in F \cap Z$. Let $r_i = (m_i, w_i, m_{i+1})$ be the first macrotransition of the suffix. The definition of AT_Z implies that there is a configuration $c \in m_{i+1} \cap \mathrm{AC}_Z$. The construction of $\mathcal{T}$ guaranties that there exists a sequence of multitransitions of $\mathcal{A}$ leading to the configuration c. More precisely, there is a sequence $T_0 T_1 \dots T_i$ such that $\mathrm{dom}(T_0)$ is an initial configuration of $\mathcal{A}$, T_j is labelled by w_j for each $0 \leq j \leq i$, $\mathrm{range}(T_j) = \mathrm{dom}(T_{j+1})$ for each $0 \leq j < i$, and $\mathrm{range}(T_i) = c$. We show that this sequence is in fact a prefix of an accepting run of $\mathcal{A}$ over w bounded by Z.

We inductively define a multitransition sequence $T_{i+1} T_{i+2} \dots$ completing this run. The definition uses the suffix $r_{i+1} r_{i+2} \dots$ of $\sigma(w)$. Let us assume that $j > i$ and that $\mathrm{range}(T_{j-1})$ is a configuration of AC_Z. We define T_j to contain one w_j-transition of s for each $s \in \mathrm{range}(T_{j-1})$. Thus we get $\mathrm{dom}(T_j) = \mathrm{range}(T_{j-1})$. As $r_j \in \mathrm{AT}_Z$, there exists a multitransition T' labelled by w_j such that both source and target configurations of T' are in AC_Z. For each must-state $s \in \mathrm{range}(T_{j-1})$, T_j contains the same transition leading from s as contained in T'. For may-states $f \in \mathrm{range}(T_{j-1})$, we have two cases. If $r_j \in \mathrm{AT}_Z^f$, T_j contains a non-looping transition leading from f to some states in Z. The existence of such a transition follows from the definition of AT_Z^f. For the remaining may-states, T_j uses selfloops. Formally, $T_j = \{t_j^s \mid s \in \mathrm{range}(T_{j-1})\}$, where

$$t_j^s = \begin{cases} (s, w_j, c_s) \text{ contained in } T' & \text{if } s \in \mathit{must}(Z) \\ (s, w_j, \{s\}) & \text{if } s \in F \wedge r_j \notin \mathrm{AT}_Z^s \\ (s, w_j, c_s) \text{ where } c_s \subseteq Z, s \notin c_s & \text{if } s \in F \wedge r_j \in \mathrm{AT}_Z^s \end{cases}$$

One can easily check that $\mathit{range}(T_j) \in \mathrm{AC}_Z$ and we continue by building T_{j+1}.

To sum up, the constructed run is bounded by Z. Moreover, T_j contains no looping transition of f whenever $r_j \in \mathrm{AT}_Z^f$. As the run $\sigma(w)$ is accepting, $r_j \in \mathrm{AT}_Z^f$ holds infinitely often for each $f \in F \cap Z$. The constructed run of $\mathcal{A}$ over w is thus accepting. □

The previous two lemmata give us the following theorem.

Theorem 1. *The TGDRA $\mathcal{G} = (\mathcal{T}, \{\mathcal{GR}_Z \mid Z \subseteq S\})$ is equivalent to $\mathcal{A}$.*

5 Translation of TGDRA to DRA

This section presents a variant of the standard degeneralization procedure. At first we illustrate the idea on a TGDRA $\mathcal{G}' = (M, \Sigma, \delta_{\mathcal{T}}, m_I, \{(K, \{L^j\}_{1 \leq j \leq h})\})$ with one generalized Rabin pair. Recall that a run is accepting if it has a suffix not using macrotransitions of K and using macrotransitions of each L^j infinitely often.

An equivalent DRA $\mathcal{D}'$ consists of $h+2$ copies of $\mathcal{G}'$. The copies are called *levels*. We start at the level 1. Intuitively, being at a level j for $1 \leq j \leq h$ means that we are waiting for a transition from L^j. Whenever a transition of K appears, we move to the level 0. A transition $r \notin K$ gets us from a level j to the maximal level $l \geq j$ such that $r \in L^{j'}$ for each $j \leq j' < l$. The levels 0 and $h+1$ have the same transitions (including target levels) as the level 1. A run of $\mathcal{G}'$ is accepting if and only if the corresponding run of $\mathcal{D}'$ visits the level 0 only finitely often and it visits the level $h+1$ infinitely often.

In general case, we track the levels for all generalized Rabin pair simultaneously. Given a TGDRA $\mathcal{G} = (M, \Sigma, \delta_{\mathcal{T}}, m_I, \{(K_i, \{L_i^j\}_{1 \leq j \leq h_i})\}_{1 \leq i \leq k})$, we construct an equivalent DRA as $\mathcal{D} = (Q, \Sigma, \delta_{\mathcal{D}}, q_i, \{(K_i', L_i')\}_{1 \leq i \leq k})$, where

- $Q = M \times \{0, 1, \ldots, h_1{+}1\} \times \cdots \times \{0, 1, \ldots, h_k{+}1\}$,
- $((m, l_1, \ldots, l_k), \alpha, (m', l_1', \ldots, l_k')) \in \delta_{\mathcal{D}}$ iff $r = (m, \alpha, m') \in \delta_{\mathcal{T}}$ and for each $1 \leq i \leq k$ it holds

$$l_i' = \begin{cases} 0 & \text{if } r \in K_i \\ \max\{l_i \leq l \leq h_i{+}1 \mid \forall l_i \leq j < l : r \in L_i^j\} & \text{if } r \notin K_i \wedge 1 \leq l_i \leq h_i \\ \max\{1 \leq l \leq h_i{+}1 \mid \forall 1 \leq j < l : r \in L_i^j\} & \text{if } r \notin K_i \wedge l_i \in \{0, h_i{+}1\}, \end{cases}$$

- $q_i = (m_I, 1, \ldots, 1)$,
- $K_i' = \{(m, l_1, \ldots, l_k) \in Q \mid l_i = 0\}$, and
- $L_i' = \{(m, l_1, \ldots, l_k) \in Q \mid l_i = h_i + 1\}$.

6 Complexity

This section discusses the upper bounds of the individual steps of our translation and compares the overall complexity to complexity of the other translations.

Given a formula φ of LTL($\mathsf{F_s}, \mathsf{G_s}$), we produce an MMAA with at most n states, where n is the length of φ. Then we build the TGDRA $\mathcal{G}$ with at most 2^{2^n} states and at most 2^n generalized Rabin pairs. To obtain the DRA $\mathcal{D}$, we multiply the state space by at most $|Z|{+}2$ for each generalized Rabin pair $\mathcal{GR}_Z$. The value of $|Z|$ is bounded by n. Altogether, we can derive an upper bound on the number of states of the resulting DRA as

$$|Q| \leq 2^{2^n} \cdot (n+2)^{2^n} = 2^{2^n} \cdot 2^{2^n \cdot \log_2(n+2)} = 2^{2^n} \cdot 2^{2^{n + \log_2 \log_2 (n+2)}} \in 2^{\mathcal{O}(2^{n + \log \log n})},$$

which is the same bound as in [19], but lower than $2^{\mathcal{O}(2^{n + \log n})}$ of `ltl2dstar`. It is worth mentioning that the number of states of our TGDRA is bounded by $2^{2^{|\varphi|}}$ while the number of states of the GDRA produced by Rabinizer is bounded by $2^{2^{|\varphi|}} \cdot 2^{AP(\varphi)}$.

7 Simplifications and Translation Improvements

An important aspect of our translation process is simplification of all intermediate results leading to smaller resulting DRA.

We simplify input formulae by reduction rules of LTL3BA, see [4] for more details. Additionally, we rewrite the subformulae of the form $\mathsf{GF}\psi$ and $\mathsf{FG}\psi$ to equivalent formulae $\mathsf{GF_s}\psi$ and $\mathsf{FG_s}\psi$ respectively. This preference of strict temporal operators often yields smaller resulting automata.

Alternating automata are simplified in the same way as in LTL2BA: removing unreachable states, merging equivalent states, and removing redundant transitions, see [15] for details.

We improve the translation of an MMAA $\mathcal{A}$ to a TGDRA $\mathcal{G}$ in order to reduce the number of generalized Rabin pairs of $\mathcal{G}$. One can observe that, for any accepting run ρ of $\mathcal{A}$, $Inf_s(\rho)$ contains only states reachable from some must-state. Hence, in the construction of acceptance condition of $\mathcal{G}$ we can consider only subsets Z of states of $\mathcal{A}$ of this form. Further, we omit a subset Z if, for each accepting run over w bounded by Z, there is also an accepting run over w bounded by some $Z' \subseteq Z$. The formal description of subsets Z considered in the construction of the TGDRA $\mathcal{G}$ is described in the full version of this paper [3].

If a run $T_0T_1\dots$ of an MMAA satisfies range$(T_i) = \emptyset$ for some i, then $T_j = \emptyset$ for all $j \geq i$ and the run is accepting. We use this observation to improve the construction of the semiautomaton $\mathcal{T}$ of the TGDRA $\mathcal{G}$: if a macrostate m contains the empty configuration, we remove all other configurations from m.

After we build the TGDRA, we simplify its acceptance condition in three ways (similar optimizations are also performed by Rabinizer).

1. We remove some generalized Rabin pairs $(K_i, \{L_i^j\}_{j\in J_i})$ that cannot be satisfied by any run, in particular when $K_i = \delta_{\mathcal{T}}$ or $L_i^j = \emptyset$ for some $j \in J_i$.
2. We remove L_i^j if there is some $l \in J_i$ such that $L_i^l \subseteq L_i^j$.
3. If the fact that a run ρ satisfies the pair $\mathcal{GR}_Z$ implies that ρ satisfies also some other pair $\mathcal{GR}_{Z'}$, we remove $\mathcal{GR}_Z$.

Finally, we simplify the state spaces of both TGDRA and DRA such that we iteratively merge the equivalent states. Two states of a DRA $\mathcal{D}$ are equivalent if they belong to the same sets of the acceptance condition of $\mathcal{D}$ and, for each α, their α-transitions lead to the same state. Two states of a TGDRA $\mathcal{G}$ are equivalent if, for each α, their α-transitions lead to the same state and belong to the same sets of the acceptance condition of $\mathcal{G}$. Moreover, if the initial state of $\mathcal{D}$ or $\mathcal{G}$ has no selfloop, we check its equivalence to another state regardless of the acceptance condition (note that a membership in acceptance condition sets is irrelevant for states or transitions that are passed at most once by any run).

Of course, we consider only the reachable state space at every step.

8 Beyond LTL(F_s,G_s) Fragment: May/Must in the Limit

The Section 4 shows a translation of MMAA into TGDRA. In fact, our translation can be used for a larger class of very weak alternating automata called *may/must in the limit automata* (limMMAA). A VWAA $\mathcal{B}$ is a limMMAA if $\mathcal{B}$ contains only must-states, states without looping transitions, and co-Büchi accepting states (not exclusively may-states), and each state reachable from a must-state is either a must- or a may-state. Note that each accepting run of a limMMAA has a suffix that contains either only empty configurations, or configurations consisting of must-states and may-states reachable from must-states. Hence, the MMAA to TGDRA translation produces correct results also for limMMAA under an additional condition: generalized Rabin pairs $\mathcal{GR}_Z$ are constructed only for sets Z that contain only must-states and may-states reachable from them.

We can obtain limMMAA by the LTL to VWAA translation of [15] when it is applied to an LTL fragment defined as

$$\varphi \ ::= \ \psi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \mathsf{X}\varphi \ \mid \ \varphi \,\mathsf{U}\, \varphi,$$

where ψ ranges over LTL($\mathsf{F_s}$,$\mathsf{G_s}$). Note that this fragment is strictly more expressive than LTL($\mathsf{F_s}$,$\mathsf{G_s}$).

9 Experimental Results

We have made an experimental implementation of our translation (referred to as *LTL3DRA*). The translation of LTL to alternating automata is taken from LTL3BA [4]. We compare the automata produced by LTL3DRA to those produced by Rabinizer and `ltl2dstar`. All the experiments are run on a Linux laptop (2.4GHz Intel Core i7, 8GB of RAM) with a timeout set to 5 minutes.

Tables given below (i) compare the sizes of the DRA produced by all the tools and (ii) show the number of states of the generalized automata produced by LTL3DRA and Rabinizer. Note that LTL3DRA uses TGDRA whereas Rabinizer uses (state-based) GDRA, hence the numbers of their states cannot be directly compared. The sizes of DRA are written as $s(r)$, where s is the number of states and r is the number of Rabin pairs. For each formula, the size of the smallest DRA (measured by the number of states and, in the case of equality, by the number of Rabin pairs) is printed in bold.

Table 1 shows the results on formulae from [14] extended with another parametric formula. For the two parametric formulae, we give all the parameter values n for which at least one tool finished before timeout. For all formulae in the table, our experimental implementation generates automata of the same or smaller size as the others. Especially in the case of parametric formulae, the automata produced by LTL3DRA are considerably smaller. We also note that the TGDRA constructed for the formulae are typically very small.

Table 2 shows the results on formulae from Spec Patterns [13] (available online[2]). We only take formulae LTL3DRA is able to work with, i.e. the formulae of the LTL fragment defined in Section 8. The fragment covers 27 out of 55 formulae listed on the web page. The dash sign in Rabinizer's column means

[2] `http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml`

Formula		LTL3DRA DRA	LTL3DRA TGDRA	Rabinizer DRA	Rabinizer GDRA	`ltl2dstar` DRA
$\mathsf{G}(a \vee \mathsf{F}b)$		**3(2)**	2	4(2)	5	4(1)
$\mathsf{FG}a \vee \mathsf{FG}b \vee \mathsf{GF}c$		**8(3)**	1	**8(3)**	8	**8(3)**
$\mathsf{F}(a \vee b)$		**2(1)**	2	**2(1)**	2	**2(1)**
$\mathsf{GF}(a \vee b)$		**2(1)**	1	**2(1)**	4	**2(1)**
$\mathsf{G}(a \vee \mathsf{F}a)$		**2(1)**	1	2(2)	2	**2(1)**
$\mathsf{G}(a \vee b \vee c)$		**2(1)**	2	**2(1)**	8	3(1)
$\mathsf{G}(a \vee \mathsf{F}(b \vee c))$		**3(2)**	2	4(2)	9	4(1)
$\mathsf{F}a \vee \mathsf{G}b$		**3(2)**	3	**3(2)**	3	4(2)
$\mathsf{G}(a \vee \mathsf{F}(b \wedge c))$		**3(2)**	2	4(2)	11	4(1)
$\mathsf{FG}a \vee \mathsf{GF}b$		**4(2)**	1	**4(2)**	4	**4(2)**
$\mathsf{GF}(a \vee b) \wedge \mathsf{GF}(b \vee c)$		**3(1)**	1	**3(1)**	8	7(2)
$(\mathsf{FF}a \wedge \mathsf{G}\neg a) \vee (\mathsf{GG}\neg a \wedge \mathsf{F}a)$		**1(0)**	1	**1(0)**	1	**1(0)**
$\mathsf{GF}a \wedge \mathsf{FG}b$		**3(1)**	1	**3(1)**	4	**3(1)**
$(\mathsf{GF}a \wedge \mathsf{FG}b) \vee (\mathsf{FG}\neg a \wedge \mathsf{GF}\neg b)$		**4(2)**	1	**4(2)**	4	5(2)
$\mathsf{FG}a \wedge \mathsf{GF}a$		**2(1)**	1	**2(1)**	2	**2(1)**
$\mathsf{G}(\mathsf{F}a \wedge \mathsf{F}b)$		**3(1)**	1	**3(1)**	4	5(1)
$\mathsf{F}a \wedge \mathsf{F}\neg a$		**4(1)**	4	**4(1)**	4	**4(1)**
$(\mathsf{G}(b \vee \mathsf{GF}a) \wedge \mathsf{G}(c \vee \mathsf{GF}\neg a)) \vee \mathsf{G}b \vee \mathsf{G}c$		**12(3)**	4	18(4)	18	13(3)
$(\mathsf{G}(b \vee \mathsf{FG}a) \wedge \mathsf{G}(c \vee \mathsf{FG}\neg a)) \vee \mathsf{G}b \vee \mathsf{G}c$		**4(2)**	4	6(3)	18	14(4)
$(\mathsf{F}(b \wedge \mathsf{FG}a) \vee \mathsf{F}(c \wedge \mathsf{FG}\neg a)) \wedge \mathsf{F}b \wedge \mathsf{F}c$		**5(2)**	4	**5(2)**	18	7(1)
$(\mathsf{F}(b \wedge \mathsf{GF}a) \vee \mathsf{F}(c \wedge \mathsf{GF}\neg a)) \wedge \mathsf{F}b \wedge \mathsf{F}c$		**5(2)**	4	**5(2)**	18	7(2)
$\mathsf{GF}(\mathsf{F}a \vee \mathsf{GF}b \vee \mathsf{FG}(a \vee b))$		**4(3)**	1	**4(3)**	4	14(4)
$\mathsf{FG}(\mathsf{F}a \vee \mathsf{GF}b \vee \mathsf{FG}(a \vee b))$		**4(3)**	1	**4(3)**	4	145(9)
$\mathsf{FG}(\mathsf{F}a \vee \mathsf{GF}b \vee \mathsf{FG}(a \vee b) \vee \mathsf{FG}b)$		**4(3)**	1	**4(3)**	4	145(9)
$\bigwedge_{i=1}^{n}(\mathsf{GF}a_i \to \mathsf{GF}b_i)$	$n=1$	**4(2)**	1	**4(2)**	4	**4(2)**
	$n=2$	**18(4)**	1	20(4)	16	11324(8)
	$n=3$	**166(8)**	1	470(8)	64	timeout
	$n=4$	**7408(16)**	1	timeout		timeout
$\bigwedge_{i=1}^{n}(\mathsf{GF}a_i \vee \mathsf{FG}a_{i+1})$	$n=1$	**4(2)**	1	**4(2)**	4	**4(2)**
	$n=2$	**10(4)**	1	11(4)	8	572(7)
	$n=3$	**36(6)**	1	52(6)	16	290046(13)
	$n=4$	**178(9)**	1	1288(9)	32	timeout
	$n=5$	**1430(14)**	1	timeout		timeout
	$n=6$	**20337(22)**	1	timeout		timeout

Table 1. The benchmark from [14] extended by one parametric formula.

that Rabinizer cannot handle the corresponding formula as it is not from the LTL(F, G) fragment. For most of the formulae in the table, LTL3DRA produces the smallest DRA. In the remaining cases, the DRA produced by our translation is only slightly bigger than the smallest one. The table also illustrates that LTL3DRA handles many (pseudo)realistic formulae not included in LTL(F, G).

Experimental results for another four parametric formulae are provided in the full version of this paper [3].

	LTL3DRA		Rabinizer		`ltl2dstar`
	DRA	TGDRA	DRA	GDRA	DRA
φ_{2}	**4(2)**	4	—		5(2)
φ_{3}	4(2)	3	4(2)	5	**4(1)**
φ_{7}	**4(2)**	3	—		**4(2)**
φ_{8}	**3(2)**	3	**3(2)**	5	4(2)
φ_{11}	**6(2)**	6	—		10(3)
φ_{12}	**8(2)**	8	—		9(2)
φ_{13}	**7(3)**	7	—		11(3)
φ_{17}	**4(2)**	4	—		5(2)
φ_{18}	4(2)	3	4(2)	5	**4(1)**
φ_{21}	**4(2)**	3	—		**4(2)**
φ_{22}	**4(2)**	4	—		5(2)
φ_{23}	**5(3)**	4	—		**5(3)**
φ_{26}	**3(2)**	2	4(2)	5	4(1)

	LTL3DRA		Rabinizer		`ltl2dstar`
	DRA	TGDRA	DRA	GDRA	DRA
φ_{27}	**4(2)**	4	—		5(2)
φ_{28}	6(3)	3	8(3)	14	**5(1)**
φ_{31}	**4(2)**	4	—		6(2)
φ_{32}	**5(2)**	5	—		7(2)
φ_{33}	**5(2)**	5	—		7(3)
φ_{36}	6(3)	4	—		**6(2)**
φ_{37}	**6(2)**	6	—		8(3)
φ_{38}	7(4)	5	—		**6(3)**
φ_{41}	**21(3)**	7	—		45(3)
φ_{42}	**12(2)**	12	—		17(2)
φ_{46}	**15(3)**	5	—		20(2)
φ_{47}	7(2)	7	—		**6(2)**
φ_{48}	**14(3)**	6	—		24(2)
φ_{52}	7(2)	7	—		**6(2)**

Table 2. The benchmark with selected formulae from SPEC PATTERNS. φ_i denotes the i-th formula on the web page.

10 Conclusion

We present another Safraless translation of an LTL fragment to deterministic Rabin automata (DRA). Our translation employs a new class of *may/must alternating automata*. We prove that the class is expressively equivalent to the LTL($\mathsf{F_s}, \mathsf{G_s}$) fragment. Experimental results show that our translation typically produces DRA of a smaller or equal size as the other two translators of LTL (i.e. Rabinizer and `ltl2dstar`) and it sometimes produces automata that are significantly smaller.

References

1. R. Alur and S. L. Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.
2. T. Babiak, T. Badie, A. Duret-Lutz, M. Křetínský, and J. Strejček. Compositional approach to suspension and other improvements to ltl translation. In *SPIN 2013*, volume 7976 of *LNCS*, pages 81–98. Springer, 2013.
3. T. Babiak, F. Blahoudek, M. Kretínský, and J. Strejcek. Effective translation of LTL to deterministic rabin automata: Beyond the (F,G)-fragment. *CoRR*, abs/1306.4636, 2013.
4. T. Babiak, M. Křetínský, V. Řehák, and J. Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *TACAS'12*, volume 7214 of *LNCS*, pages 95–109. Springer, 2012.
5. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
6. K. Chatterjee, A. Gaiser, and J. Křetínský. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In *CAV'13*, volume 8044 of *LNCS*, pages 559–575. Springer, 2013.

7. A. Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1962.
8. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 1999.
9. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.
10. J.-M. Couvreur. On-the-fly verification of temporal logic. In *FM'99*, volume 1708 of *LNCS*, pages 253–271. Springer, 1999.
11. M. Daniele, F. Giunchiglia, and M. Y. Vardi. Improved automata generation for linear temporal logic. In *CAV '99*, volume 1633 of *LNCS*, pages 249–260. Springer, 1999.
12. A. Duret-Lutz. LTL translation improvements in Spot. In *VECoS'11*, Electronic Workshops in Computing. British Computer Society, 2011.
13. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *ICSE 1999*, pages 411–420. IEEE, 1999.
14. A. Gaiser, J. Křetínský, and J. Esparza. Rabinizer: Small deterministic automata for LTL(F,G). In *ATVA'12*, volume 7561 of *LNCS*, pages 72–76, 2012.
15. P. Gastin and D. Oddoux. Fast LTL to Büchi Automata Translation. In *CAV'01*, volume 2102 of *LNCS*, pages 53–65. Springer, 2001.
16. J. Klein. ltl2dstar – LTL to deterministic Streett and Rabin automata. http://www.ltl2dstar.de.
17. J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2):182–195, 2006.
18. J. Klein and C. Baier. On-the-fly stuttering in the construction of deterministic ω-automata. In *CIAA 2007*, volume 4783 of *LNCS*, pages 51–61. Springer, 2007.
19. J. Křetínský and J. Esparza. Deterministic automata for the (F, G)-fragment of LTL. In *CAV'12*, volume 7358 of *LNCS*, pages 7–22. Springer, 2012.
20. O. Kupferman. Recent challenges and ideas in temporal synthesis. In *SOFSEM 2012*, volume 7147 of *LNCS*, pages 88–98. Springer, 2012.
21. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.
22. A. Morgenstern and K. Schneider. From LTL to symbolically represented deterministic automata. In *VMCAI 2008*, volume 4905 of *LNCS*, pages 279–293. Springer, 2008.
23. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.
24. N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. In *VMCAI'06*, volume 3855 of *LNCS*, pages 364–380. Springer, 2006.
25. A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57. IEEE, 1977.
26. A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *ICALP'89*, volume 372 of *LNCS*, pages 652–671. Springer, 1989.
27. S. Safra. On the complexity of omega-automata. In *FOCS'88*, pages 319–327. IEEE Computer Society, 1988.
28. S. Schewe. Tighter bounds for the determinisation of Büchi automata. In *FOSSACS'09*, volume 5504 of *LNCS*, pages 167–181. Springer, 2009.
29. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *CAV'00*, volume 1855 of *LNCS*, pages 248–263. Springer, 2000.
30. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS'85*, pages 327–338. IEEE Computer Society, 1985.
31. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS'86*, pages 332–344. IEEE Computer Society, 1986.