Limit Deterministic and Probabilistic Automata for LTL\GU

Dileep Kini and Mahesh Viswanathan

Department of Computer Science,
University of Illinois at Urbana-Champaign, Urbana, IL, USA

Abstract. LTL\$GU$ is a fragment of linear temporal logic (LTL), where negations appear only on propositions, and formulas are built using the temporal operators $\boldsymbol{X}$ (next), $\boldsymbol{F}$ (eventually), $\boldsymbol{G}$ (always), and $\boldsymbol{U}$ (until, with the restriction that no until operator occurs in the scope of an always operator. Our main result is the construction of Limit Deterministic Büchi automata for this logic that are exponential in the size of the formula. One consequence of our construction is a new, improved EXPTIME model checking algorithm (as opposed to the previously known doubly exponential time) for Markov Decision Processes and LTL\$GU$ formulae. Another consequence is that it gives us a way to construct exponential sized Probabilistic Büchi Automata for LTL\$GU$.

1 Introduction

Starting with the seminal work of Vardi, Wolper, and Sistla [17], there has been a lot of interest in discovering efficient translations of Linear Temporal logic (LTL) formulae into small automata (see [16,7,10,14,13,11,2,6] for example). The reason for this is that logic to automata translations have a direct impact on algorithms of verification and synthesis of systems [18]. When verifying systems, one is often satisfied with constructing nondeterministic Büchi automata for LTL formulae. However, for a couple of applications, general nondeterministic automata don't suffice — when synthesizing reactive modules for LTL specifications, deterministic automata are necessary, and when model checking Markov Decision Processes with respect to almost sure satisfaction of LTL specifications, one needs either deterministic or limit deterministic automata. As a consequence, a series of papers recently present algorithms and tools for constructing deterministic automata from LTL specifications [10,14,13,11,12,2,6]; though in the worst case the size of the constructed automata are doubly exponential, these algorithms have been shown to construct small automata for a number of examples. In this paper, we investigate whether there are provable improvements in translating fragments of LTL to limit deterministic automata, and explore whether these can then be exploited to improve the asymptotic complexity of the MDP model checking problem.

We consider the fragment LTL\GU, first introduced in [12]. In this logic, formulae are built from propositions and their negations using conjunction, disjunction, and the temporal operators $\boldsymbol{X}$ (next), $\boldsymbol{F}$(eventually), $\boldsymbol{G}$ (always), and

C. Baier and C. Tinelli (Eds.): TACAS 2015, LNCS 9035, pp. 628–642, 2015.
DOI: 10.1007/978-3-662-46681-0_57

$\boldsymbol{U}$ (until), with the restriction that no $\boldsymbol{U}$ operator appears in the scope of a $\boldsymbol{G}$ operator. Our main result is a translation of LTL\$GU$ formulae into nondeterministic Büchi automata of exponential size that is *deterministic in the limit* — an automaton is deterministic in the limit (or limit deterministic) if the transitions from any state that is reachable from an accepting state are deterministic. This construction should be contrasted with the observation that any translation from LTL\$GU$ to deterministic automata must in the worst case result in automata that are doubly exponential in size [1]; in fact, this lower bound applies to any fragment of LTL that has $\vee$, $\wedge$, and $\boldsymbol{F}$.

Our construction of limit deterministic automata for LTL\$GU$ proceeds in two steps. First we construct limit deterministic automata for LTL(F,G) which is the LTL fragment without until, i.e., with just the temporal operators next, always, and eventually. Next, we observe that the automaton for $\varphi \in$ LTL\$GU$ can be seen as the composition of two limit deterministic automata: one automata for the formula ψ, where all the until-free subformulae of φ are replaced by propositions, and another automaton for the until-free subformulae of φ. This composition is reminiscent of the master-slave composition in [6] and the composition of temporal testers [15] but with some differences.

Our construction of exponentially sized limit deterministic automata for LTL\$GU$ has complexity theoretic consequences for model checking MDPs. Courcoubetis and Yannakakis [5] proved that the problem of model checking MDPs against LTL is 2EXPTIME-complete. Our automata construction, coupled with the algorithm outlined in [5], shows that model checking MDPs against LTL\$GU$ is in EXPTIME; we prove a matching lower bound in this paper as well. Thus, for a large, expressively rich subclass of LTL specifications, our results provide an exponential improvement to the complexity of model checking MDPs.

Another consequence of our main result is that it gives us a way to translate LTL\GU formulae to exponential sized probabilistic Büchi automata (PBA) [3]. Probabilistic Büchi automata are like Büchi automata, except that they probabilistically choose the next state on reading an input symbol. On input w, such a machine can be seen as defining a probability measure on the space of all runs on w. A PBA is said to accept a (infinite length) string w iff the set of all accepting runs (i.e., runs that visit some final state infinitely often) have measure > 0. We use the observation that any assignment of non-zero probabilities to the nondeterministic choices of a limit deterministic NBA, results in a PBA that accepts the same language [3]. This result also generalizes some of the results in [8] where exponential sized *weak probabilistic monitors* [1] are constructed for the LTL fragment with just the temporal operators $\boldsymbol{X}$ and $\boldsymbol{G}$.

The rest of the paper is organized as follows. In Section 2 we introduce the notations and definitions we use in the paper. In Section 3 we present a translation from LTL(F,G) to limit deterministic NBAs. In Section 4 we give a compositional style construction for formulae in LTL\GU by using the construction in

[1] A weak finite state probabilistic monitor [4] is a PBA with the restriction that all states except a unique reject state are final, and all transitions from the unique rejecting state are self loops.

the previous section. In Section 5 we reflect on the consequences of our results and finally give concluding remarks in Section 6.

2 Preliminaries

First we introduce the notation we use throughout the paper. We use P to denote the set of propositions. An assignment ν is a function mapping all propositions to true or false. We use w to denote infinite words over a finite alphabet. We use $w[i]$ to denote the i^{th} symbol in the sequence w, and use w_i to denote the suffix $w[i]w[i+1]\ldots$ of w starting at i. We use $[n]$ to denote all non-negative integers less than n that is $\{0, 1, \ldots, n-1\}$. We shall use Σ, Γ to denote finite sets of symbols.

Definition 1 (Syntax). *The formulae in the fragment* LTL(F,G) *over* P *is given by the following syntax*

$$\varphi \quad ::= \quad p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \boldsymbol{X}\varphi \mid \boldsymbol{F}\varphi \mid \boldsymbol{G}\varphi \qquad p \in P$$

and the formulae in the fragment LTL$\backslash GU$ *are given by the syntax*

$$\psi \quad ::= \quad \varphi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \boldsymbol{X}\psi \mid \psi \,\boldsymbol{U}\psi \qquad \varphi \in \text{LTL}(F,G)$$

Let $\mathbb{F}_\varphi$ and $\mathbb{G}_\varphi$ denote the set of all $\boldsymbol{F}$ and $\boldsymbol{G}$ subformulae of φ respectively. We drop the subscript whenever the formula φ is clear from the context. For a temporal operator $Op \in \{\boldsymbol{G}, \boldsymbol{F}, \boldsymbol{U}\}$ we use LTL(Op) to denote the fragment consisting of formulae built using $Op, \boldsymbol{X}, \vee, \wedge$, and $\neg$ with negation allowed to appear only on propositions.

Definition 2 (Semantics). *LTL formulae over a set* P *are interpreted over words* w *in* $(2^P)^\omega$. *The semantics of the logic are given by the following rules*

$$\begin{array}{lll@{\qquad}lll}
w \vDash p & \iff & p \in w[0] & w \vDash \boldsymbol{X}\varphi & \iff & w_1 \vDash \varphi \\
w \vDash \neg p & \iff & p \notin w[0] & w \vDash \boldsymbol{F}\varphi & \iff & \exists\, i : w_i \vDash \varphi \\
w \vDash \varphi \wedge \psi & \iff & w \vDash \varphi \text{ and } w \vDash \psi & w \vDash \boldsymbol{G}\varphi & \iff & \forall\, i : w_i \vDash \varphi \\
w \vDash \varphi \vee \psi & \iff & w \vDash \varphi \text{ or } w \vDash \psi & w \vDash \varphi \,\boldsymbol{U}\psi & \iff & \exists\, i : w_i \vDash \psi, \text{and} \\
 & & & & & \quad \forall\, j < i : w_j \vDash \varphi
\end{array}$$

The semantics of φ, *denoted by* $[\![\varphi]\!]$, *is defined as the set* $\{w \in (2^P)^\omega \mid w \vDash \varphi\}$.

Definition 3 (Büchi Automata). *A nondeterministic Büchi automaton (NBA) over input alphabet* Σ *is a tuple* (Q, δ, I, F) *where* Q *is a finite set of states;* $\delta \subseteq Q \times \Sigma \times Q$ *is a set of transitions;* $I \subseteq \delta$ *is a set of initial transitions* [2] *and* F *is a set of final states. We say state* $q \in Q$ *is initial if* $(q, \sigma, q') \in I$ *for some* σ *and* q'.

[2] We use initial transitions instead of states for notational convenience. It can be easily converted into a state based definition.

A run of a NBA over a word $w \in \Sigma^\omega$ *is an infinite sequence of states* $q_0 q_1 q_2 \ldots$ *such that* $(q_0, w[0], q_1) \in I$ *and* $\forall\, i \geq 0\ (q_i, w[i], q_{i+1}) \in \delta$. *A run is accepting if* $q_i \in F$ *for infinitely many* i.

The language accepted by an NBA $\mathcal{A}$, *denoted by* $L(\mathcal{A})$ *is the set of all words* $w \in \Sigma^\omega$ *which have an accepting run on* $\mathcal{A}$.

Definition 4 (Limit Determinism). *A NBA* (Q, δ, I, F) *over input alphabet* Σ *is said to be* limit deterministic *if for every state* q *reachable from a final state, it is the case that* $|\delta(q, \sigma)| \leq 1$ *for every* $\sigma \in \Sigma$.

We make note of the fact that any limit deterministic NBA translation for a fragment devoid of the $\boldsymbol{X}$ operator can be converted into translation for the fragment with $\boldsymbol{X}$ by incurring a multiplicative factor blow-up that is exponential in the number of nested $\boldsymbol{X}$s in the formula. A proof sketch of this result can be found in the companion technical report [9].

Proposition 5. *Let* LTL$'$ *be some fragment of* LTL. *If for every* $\varphi \in \text{LTL}'\backslash\boldsymbol{X}$ *one can build a limit deterministic NBA* $\mathcal{A}_\varphi$ *such that it recognizes* $[\![\varphi]\!]$ *and is of size* $f(|\varphi|)$, *then for every* $\varphi' \in \text{LTL}'$ *one can build a limit deterministic NBA* $\mathcal{A}'_{\varphi'}$ *such that it recognizes* $[\![\varphi']\!]$ *and has size* $\mathcal{O}(2^n f(|\varphi'|))$ *where* n *is the number of nested* $\boldsymbol{X}$*s appearing in* φ'.

The compositional construction for LTL\GU requires we deal with automata with outputs. For this purpose we define Mealy automata with Büchi acceptance which we use in Section 4.

Definition 6 (Mealy Automata). *A nondeterministic Mealy machine with Büchi acceptance (NBM) with input alphabet* Σ *and output alphabet* Γ *is a tuple* (Q, δ, I, M, F) *where* (Q, δ, I, F) *is an NBA with input alphabet* Σ *and* $M : Q \times \Sigma \to \Gamma$ *is a partial function that is defined on all* (q, σ) *for which there is a* q' *such that* $(q, \sigma, q') \in \delta$.

The relation accepted by an NBM is the set of all pairs $(w, \lambda) \in \Sigma^\omega \times \Gamma^\omega$ *such that* w *is accepted by the NBA* (Q, δ, I, F) *and* λ *is such that there is an accepting run of* w *of the form* $q_0 q_1 q_2 \ldots$ *where* $M(q_i, w[i]) = \lambda[i]$ *for all* i.

In section 5 we describe our result regarding construction of probabilistic Büchi automata, which we define next.

Definition 7 (Probabilistic Automata). *A probabilistic Büchi automaton (PBA) over input alphabet* Σ *is a tuple* (Q, Δ, q_s, F) *where* Q *is a finite set of states;* $\Delta : Q \times \Sigma \times Q \to [0, 1]$ *specifies transition probabilities such that for every* $q \in Q$ *and* $\sigma \in \Sigma$ *we have* $\sum_{r \in Q} \Delta(q, \sigma, r) = 1$; $q_s \in Q$ *is an initial state;* $F \subseteq Q$ *is a set of final states.*

Given a word $w \in \Sigma^\omega$ a PBA $\mathcal{M}$ behaves as follows: it is initially at state $q_0 = q_s$. After having seen the first i symbols $w[0]w[1] \ldots w[i-1]$ it is in state q_i. On seeing $w[i]$ it chooses the next state q_{i+1} with probability $\Delta(q_i, w[i], q_{i+1})$. It continues this process to produce a run $\rho \in Q^\omega$. A run ρ is accepting if some final state appears infinitely often.

A word w produces a Markov chain C obtained by unfolding the PBA $\mathcal{M}$ along the symbols of w [3]. The probability measure induced by this Markov chain on runs in Q^ω is used to define the acceptance probability of the word w on $\mathcal{M}$ as

$$\Pr(w) = \Pr\{\rho \in Q^\omega \mid \rho \text{ is accepting for } w \text{ over } \mathcal{M}\}$$

The language accepted by a PBA over input Σ denoted by $L_{>0}(\mathcal{M})$ is the set of all words $w \in \Sigma^\omega$ with positive acceptance probability, i.e $\Pr(w) > 0$.

We conclude this section by observing a result from [3] that enables translation from limit deterministic automata to PBAs.

Lemma 8. *[3] Given a limit deterministic NBA $\mathcal{D} = (Q, \delta, I, F)$ there is a PBA $\mathcal{M}$ with $|Q| + 1$ states such that $L(\mathcal{D}) = L_{>0}(\mathcal{M})$.*

3 Automata for LTL($\boldsymbol{F}$,$\boldsymbol{G}$) Formulae

In this section, we present a construction of exponential sized limit deterministic NBA for the fragment LTL(F,G). Thanks to Proposition 5 we ignore the $\boldsymbol{X}$ operator since we are aiming to construct exponential size automata.

The following proposition embodies the key idea behind our construction. A proof is provided in the technical report [9].

Proposition 9. *For any formula $\varphi \in$ LTL over P, and any word $w \in (2^P)^\omega$ exactly one of the following three holds*

$$w \vDash \neg \boldsymbol{F}\varphi, \quad w \vDash (\neg \boldsymbol{G}\varphi \wedge \boldsymbol{F}\varphi), \quad w \vDash \boldsymbol{G}\varphi$$

Furthermore, if φ is of the form $\boldsymbol{F}\psi$ or $\boldsymbol{G}\psi$ then we can deduce if $w \vDash \varphi$ holds from knowing which one of the above three holds.

The essence of our construction is in guessing which one of the three formulae in Proposition 9 holds for $\boldsymbol{F}$ and $\boldsymbol{G}$ subformulae. We define a *guess*, to be tripartition $\pi = \langle \pi^\tau \mid \pi^\upsilon \mid \pi^\kappa \rangle$ of $\mathbb{F}_\varphi \cup \mathbb{G}_\varphi$. Let Π denote the set of all tripartitions π of $\mathbb{F}_\varphi \cup \mathbb{G}_\varphi$. If a subformula $\boldsymbol{F}\psi$ is present in π^τ, our guess is that $\neg \boldsymbol{F}\psi (\equiv \neg \boldsymbol{F}\boldsymbol{F}\psi)$ holds for the input to be seen. If a subformula $\boldsymbol{G}\psi$ is in π^τ, our guess is that $\boldsymbol{G}\psi (\equiv \boldsymbol{G}\boldsymbol{G}\psi)$ holds for the input to be seen. Table 1 summarizes how we interpret a tripartition as one of three guesses for $\boldsymbol{F}$ and $\boldsymbol{G}$ subformulae.

Table 1. Guess corresponding to a tripartition π

	π^τ	π^υ	π^κ
$\boldsymbol{F}\psi$	$\neg \boldsymbol{F}\psi$	$\boldsymbol{F}\psi \wedge \neg \boldsymbol{G}\boldsymbol{F}\psi$	$\boldsymbol{G}\boldsymbol{F}\psi$
$\boldsymbol{G}\psi$	$\boldsymbol{G}\psi$	$\neg \boldsymbol{G}\psi \wedge \boldsymbol{F}\boldsymbol{G}\psi$	$\neg \boldsymbol{F}\boldsymbol{G}\psi$

From the second part of Proposition 9 we know exactly which formulae in $\mathbb{F}_\varphi \cup \mathbb{G}_\varphi$ are presently true according to $\pi \in \Pi$. This information along with the

knowledge of the truth of propositions at present allows us to *evaluate* the truth of the formula φ according to π. We define what it means to evaluate a formula.

Definition 10 (Evaluation). *For any formula $\varphi \in \mathrm{LTL}(F,G)$ over P, a partition $\pi \in \Pi$ and assignment ν on P we inductively define a boolean function $[\varphi]^\pi_\nu$, the evaluation of a formula φ, as follows:*

$$[p]^\pi_\nu = \nu(p) \qquad [\varphi \wedge \psi]^\pi_\nu = [\varphi]^\pi_\nu \wedge [\psi]^\pi_\nu \qquad [\boldsymbol{G}\psi]^\pi_\nu = \mathit{true}\ \mathit{iff}\ \boldsymbol{G}\psi \in \pi^\tau$$
$$[\neg p]^\pi_\nu = \neg\nu(p) \qquad [\varphi \vee \psi]^\pi_\nu = [\varphi]^\pi_\nu \vee [\psi]^\pi_\nu \qquad [\boldsymbol{F}\psi]^\pi_\nu = \mathit{true}\ \mathit{iff}\ \boldsymbol{F}\psi \notin \pi^\tau$$

Next, we observe that if π is sound in the sense that every formula $\boldsymbol{G}\psi \in \pi^\tau$ and $\boldsymbol{F}\psi \notin \pi^\tau$ is true at present then φ evaluates to true with respect to π indicates that φ is indeed true.

Proposition 11 (Soundness). *For any formula $\varphi \in \mathrm{LTL}(F,G)$, a guess $\pi \in \Pi$ and word $w \in \Sigma^\omega$ if the following implications hold*

$$\boldsymbol{G}\psi \in \pi^\tau \implies w \vDash \boldsymbol{G}\psi \qquad \boldsymbol{F}\psi \notin \pi^\tau \implies w \vDash \boldsymbol{F}\psi$$

for every $\boldsymbol{G}\psi$, $\boldsymbol{F}\psi$ that is a subformula of φ then: $[\varphi]^\pi_{w[0]} = \mathrm{true}$ implies $w \vDash \varphi$.

Proof. Induction on the structure of φ. □

Similarly, if π is complete in the sense that every $\boldsymbol{G}\psi$ that is true is in π^τ and every $\boldsymbol{F}\psi$ that is true is not in π^τ then φ is true implies it evaluates to true.

Proposition 12 (Completeness). *For any formula $\varphi \in \mathrm{LTL}(F,G)$, a guess $\pi \in \Pi$ and a word $w \in \Sigma^\omega$ if the following implications holds*

$$w \vDash \boldsymbol{G}\psi \implies \boldsymbol{G}\psi \in \pi^\tau \qquad w \vDash \boldsymbol{F}\psi \implies \boldsymbol{F}\psi \notin \pi^\tau$$

for every $\boldsymbol{G}\psi$, $\boldsymbol{F}\psi$ that is a subformula of φ then: $w \vDash \varphi$ implies $[\varphi]^\pi_{w[0]} = \mathit{true}$.

Proof. Induction on the structure of φ. □

In our construction, every state of the automaton holds a tripartition that corresponds to a guess. According to this guess some subformulae may hold now, in the future, or never. Each initial transition is such that it enables the main formula to be true. Each transition ensures the propagation of temporal requirements to successive guesses. In an accepting run we ensure that each of our guesses is sound.

Since our formulae are in negation normal form, it only makes sense to take care of checking a guess (or a part of it) on a subformula φ if the guess claims φ to be true at present or some point in the future. If a guess claims that φ does not hold we don't need to bother checking it, because even if it did it could not make a superformula false that was otherwise true. For instance, if $\boldsymbol{F}\psi \in \pi^\tau$ we don't need to enforce that $\neg\boldsymbol{F}\psi$ holds, and if $\boldsymbol{F}\psi \in \pi^\upsilon$ we only need to ensure $\boldsymbol{F}\psi$ holds but don't need to enforce $\neg\boldsymbol{G}\boldsymbol{F}\psi$. Similarly if $\boldsymbol{G}\psi \in \pi^\kappa$ then we don't

need to check $\neg \boldsymbol{F}\,\boldsymbol{G}\psi$, and if $\boldsymbol{G}\psi \in \pi^{\upsilon}$ then we only need to check $\boldsymbol{F}\,\boldsymbol{G}\psi$ but not $\neg \boldsymbol{F}\,\boldsymbol{G}\psi$.

Say $\boldsymbol{G}\psi \in \pi^{\tau}$, it requires that $\boldsymbol{G}\psi$ holds now. This can be checked by ensuring ψ holds now and $\boldsymbol{G}\psi$ holds at the next time step. We can check if ψ is true at present by evaluating ψ with respect to our guess and the incoming input symbol. We can ensure ψ holds at the next time step by propagating $\boldsymbol{G}\psi$ to the π^{τ} in the next guess. If $\boldsymbol{G}\psi \in \pi^{\upsilon}$, we need to check that $\boldsymbol{F}\,\boldsymbol{G}\psi$ holds by having $\boldsymbol{G}\psi$ either in π^{τ} or π^{υ} of the next guess. If $\boldsymbol{G}\psi$ is not moved to π^{τ} but kept in π^{υ} forever then $\boldsymbol{F}\,\boldsymbol{G}\psi$ might not hold. We overcome this by imposing an acceptance condition which requires the set π^{υ} to eventually become empty.

For $\boldsymbol{F}\psi \in \pi^{\upsilon}$ we need to ensure $\boldsymbol{F}\psi$ holds. That is either ψ should hold now or some point in the future. The former can be checked by evaluating ψ using the current guess and the incoming input symbol. If it does not evaluate to true we require $\boldsymbol{F}\psi$ to hold at the next time step, which can be ensured by having $\boldsymbol{F}\psi$ in π^{υ} in the next step. Like before, this creates a possibility for $\boldsymbol{F}\psi$ to be false if it keeps getting delayed by its presence in π^{υ} forever, but as mentioned above our acceptance condition will be such that π^{υ} is eventually empty. For $\boldsymbol{F}\psi \in \pi^{\kappa}$ we are claiming $\boldsymbol{G}\boldsymbol{F}\psi$, this can be ensured by evaluating ψ with the respect to the current guess and input symbol, and requiring the resulting valuation to be true infinitely often. This can be achieved by fixing $\boldsymbol{F}\psi$ to be in π^{κ} forever and using an appropriate Büchi condition to verify that for $\boldsymbol{F}\psi$ in π^{κ}, ψ holds infinitely often.

Before we present the formal details of the construction we illustrate the above ideas above using a simple example.

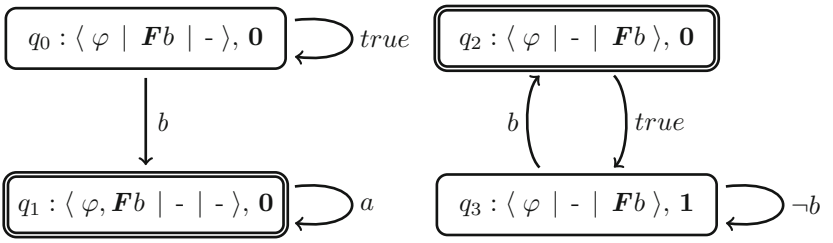

Fig. 1. The limit deterministic NBA for $\varphi = \boldsymbol{G}(a \vee \boldsymbol{F}b)$ obtained using our construction. Here states q_0, q_1, q_2 are initial with each transition going out of them being an initial transition. The final states q_1, q_2 are double bordered.

Example 13. Figure 1 shows the limit deterministic automaton constructed using our translation for the formula $\varphi = \boldsymbol{G}(a \vee \boldsymbol{F}b)$. Each state consists of a guess-counter pair (π, n). A guess here is a tripartition $\pi = \langle\, \pi^{\tau} \mid \pi^{\upsilon} \mid \pi^{\kappa} \,\rangle$ of $\{\varphi, \boldsymbol{F}b\}$, the set of all $\boldsymbol{G}, \boldsymbol{F}$ subformulae of φ. A dash '-' in the tripartition indicates that the corresponding part is empty. For simplicity the transitions are annotated by propositional formulae: a transition of the form (q, ϕ, q') indicates there is a transition from q to q' on each assignment that makes ϕ true.

(Example contd.) A transition $q \xrightarrow{\nu} q'$ is initial when the main formula φ evaluates to true on the guess in q and input ν, and the counter is 0 in q. The formula φ being a $\boldsymbol{G}$ subformula evaluates to true iff $\varphi \in \pi^{\tau}$ (see Definition 10). Hence every initial state is required to have $\varphi \in \pi^{\tau}$, and in this case because φ evaluates to true irrespective of the next input symbol every transition going out of an initial state is an initial transition.

For each $\boldsymbol{G}\psi$ formula in π^{τ} one needs to check $\boldsymbol{G}\psi$ holds for all accepting runs starting from that state. This is ensured by proceeding from guess π_1 to π_2 on input ν only if ψ evaluates to true on π_1 and ν (ensuring ψ holds now), and $\boldsymbol{G}\psi$ is present in π_2 (propagating the temporal requirement $\boldsymbol{G}\psi$ to the next state). In our example since $\varphi \in \pi^{\tau}$ for every initial state it is also propagated to every successive state, ensuring that $\varphi \in \pi^{\tau}$ for all states as shown in Figure 1. Therefore we have no freedom in the assignment of φ in our state space.

What remains is to assign the subformula $\boldsymbol{F}b$ to one of the three partitions. Consider q_0 in which it is assigned to π^{υ}. The automaton has to ensure $\boldsymbol{F}b$ holds for all accepting runs from q_0. If b holds now then it can either move to q_0 or q_1 depending upon whether it chooses to guess if $\boldsymbol{F}b$ holds again at the next time step. If b does not hold it has no choice but to remain in q_0 waiting for b to become true. The presence of $\boldsymbol{F}b \in \pi^{\upsilon}$ for q_0 ensures $(a \vee \boldsymbol{F}b)$ evaluates to true (independent of the input) thus the requirement of $\varphi \in \pi^{\tau}$ mentioned above is satisfied. Now consider q_1, here we assume that $\boldsymbol{F}b$ is false (since $\boldsymbol{F}b \in \pi^{\tau}$) and hence have to rely on a being true for $(a \vee \boldsymbol{F}b)$ to evaluate to true for the requirement of $\varphi \in \pi^{\tau}$, giving rise to the transition (q_1, a, q_1). In q_0 and q_1 the counter is not needed due to the fact that π^{κ} is empty. State q_1 is marked as final because π^{υ} is empty and the counter is 0. Now, consider q_2 and q_3 which have the same guess with $\boldsymbol{F}b \in \pi^{\kappa}$ but different counters. Since $\boldsymbol{F}b \in \pi^{\kappa}$ the formula $(a \vee \boldsymbol{F}b)$ evaluates to true irrespective of the input, thus satisfying the requirement of $\varphi \in \pi^{\tau}$. For $\boldsymbol{F}b \in \pi^{\kappa}$ we need to ensure that $\boldsymbol{GF}b$ holds, this is done by the Büchi condition which requires q_2, a final state (where π^{υ} is empty and counter is 0), to be visited infinitely often thus making sure that b becomes true infinitely often.

Next we provide the formal details of our construction for an arbitrary LTL(F,G) formula.

Definition 14 (Construction). *Given a formula φ in* LTL(F,G) *defined over propositions P, let $\mathcal{D}(\varphi)$ be the NBA (Q, I, δ, F) over the alphabet 2^P defined as follows*

- *Q is the set $\Pi \times [z]$, consisting of guess-counter pairs where $z = |\mathbb{F}_{\varphi}|+1$*
- *δ is the set of all transitions*

$$(\pi_1, m) \xrightarrow{\nu} (\pi_2, n)$$

such that
(A) for each $\boldsymbol{G}\psi \in \pi_1^{\tau}$, $[\,\psi\,]_{\nu}^{\pi_1}$ is true
(B) for each $\boldsymbol{F}\psi \in \pi_1^{\upsilon}$, $[\,\psi\,]_{\nu}^{\pi_1}$ is false implies $\boldsymbol{F}\psi \in \pi_2^{\upsilon}$.

(C) $\pi_1^\tau \subseteq \pi_2^\tau$ *and* $\pi_1^\kappa = \pi_2^\kappa$

(D) n *is updated as follows*

$$n = \begin{cases} m, & (|\pi_1^\upsilon| > 0) \vee (m > 0 \wedge \neg[\,\psi_m\,]_\nu^{\pi_1}) \\ m{+}1 \pmod k & \text{otherwise} \end{cases}$$

where $k = |\pi^\kappa \cap \mathbb{F}_\varphi| + 1$ *and* ψ_m *be such that* $\boldsymbol{F}(\psi_m)$ *is the* m^{th} *formula in* $\pi^\kappa \cap \mathbb{F}_\varphi$

- I *is the set of transitions of the form* $(\pi, 0) \xrightarrow{\nu} (\pi', i)$ *where* $[\,\varphi\,]_\nu^\pi$ *is true*
- F *is the set of states* $(\pi, 0)$ *where* π^υ *is empty.*

Next, we present the theorem that states the correctness of the above construction.

Theorem 15. *For any formula* $\varphi \in \mathrm{LTL}(F,G)$*, the NBA* $\mathcal{D}(\varphi)$ *is a limit deterministic automaton of size* $2^{\mathcal{O}(|\varphi|)}$ *such that* $L(\mathcal{D}(\varphi)) = [\![\varphi]\!]$*.*

Proof. The number of states in $\mathcal{D}(\varphi)$ is bounded by $3^{|\mathbb{F}\cup\mathbb{G}|} \times |\mathbb{F}|$ and so clearly the size of $\mathcal{D}(\varphi)$ is exponential in $|\varphi|$.

We can see that $\mathcal{D}(\varphi)$ is limit deterministic as follows: The final states are of the form $(\pi, 0)$ where π^υ is empty. Note that according to condition (C), π^υ remains empty once it becomes empty, and π^τ and π^κ remain fixed. Hence the guess π can never change after visiting a final state. And since the counter is updated deterministically we have that any state reachable from a final state chooses its next state deterministically.

The proof of the fact $L(\mathcal{D}(\varphi)) = [\![\varphi]\!]$ is provided in the companion technical report [9]. □

4 Automata for LTL\GU Formulae

In this section, we present a construction of limit deterministic NBAs for the fragment LTL\GU of exponential size. We follow a compositional approach where we compose a *master* and a *slave* automata (terminology borrowed from [6]) to obtain the required one. The master automaton assumes that the truth values of the maximal until-free subformulae are known at each step and checks whether the top-level until formula holds true. The master automaton works over an extended set of propositions where the new propositions are introduced in place of the until-free subformulae. The slave automaton works over the original set of propositions and outputs at each step the truth value of the subformulae abstracted by the master in the form of the new propositions. The master and the slave are then composed such that they work together to check the entire formula. Once again we apply Proposition 5 to ignore $\boldsymbol{X}$ operators when presenting our construction.

We first clarify some notation we use in this section. For a finite set of formulae Φ we use P_Φ to denote a set of propositions p_ϕ indexed by formulae $\phi \in \Phi$. We will use $\varphi_{/\Phi}$ to denote the formula obtained from φ by replacing subformulae of φ appearing in Φ by their corresponding propositions in P_Φ.

Definition 16 (Characteristic Relation). *For a finite set of* LTL *formulae* Φ *over propositions* P *we define its* characteristic relation $R_\Phi \subseteq (2^P)^\omega \times (2^{P_\Phi})^\omega$ *as follows:*

$$(w, \lambda) \in R_\Phi \quad \textit{iff} \quad \lambda[i] = \{p_\varphi \mid \varphi \in \Phi,\ w_i \vDash \varphi\}$$

Given $w_1 \in 2^{P_1}$ and $w_2 \in 2^{P_2}$ define the join of w_1 and w_2 denoted by $w_1 \cup w_2$ as the word in $(2^{P_1 \cup P_2})^\omega$ whose i^{th} element $(w_1 \cup w_2)[i]$ is $w_1[i] \cup w_2[i]$. Given a relation $R \subseteq (2^{P_1})^\omega \times (2^{P_2})^\omega$ define the language $R^\circ \subseteq (2^{P_1 \cup P_2})^\omega$ as the set of words obtained by joining the pairs in R:

$$R^\circ = \left\{\rho \in (2^{P_1 \cup P_2})^\omega \mid \exists (w, \lambda) \in R,\ \ \rho = w \cup \lambda\right\}$$

Later on in this section (Proposition 23) we show how to construct a NBM for a set of LTL(F,G) formula which accepts a relation that is not the characteristic relation but is "subsumed" by the characteristic relation of the set. In that direction we define what it means for a relation to be subsumed by another.

Definition 17 (Subsumption). *For two relations* $R, S \subseteq (2^{P_1})^\omega \times (2^{P_2})^\omega$ *we say that* R *is* subsumed *by* S*, denoted by* $R \triangleleft S$ *iff* $S \subseteq R$ *and for every* $(w, \lambda) \in R$ *there exists* $(w, \lambda') \in S$ *such that* $\forall i\ \lambda[i] \subseteq \lambda'[i]$.

Next, we describe how to break an LTL\GU formula into an until factor and an until-free factor which are going to be handled by the master and slave automata respectively.

Definition 18 (Factorization). *Given a formula* $\varphi \in$ LTL\GU *over propositions* P *we identify the set of maximal subformulae of* φ *that do not contain* $\boldsymbol{U}$ *as the* until-free factor *of* φ *denoted by* $\Upsilon(\varphi)$.

$$\Upsilon(\boldsymbol{G}\varphi) = \{\boldsymbol{G}\varphi\} \qquad \Upsilon(\boldsymbol{F}\varphi) = \{\boldsymbol{F}\varphi\}$$

$$\Upsilon(\varphi_1 \wedge/\vee\ \varphi_2) = \begin{cases} \{\varphi_1 \wedge/\vee\ \varphi_2\} & \textit{if } \varphi_1 \in \Upsilon(\varphi_1) \textit{ and } \varphi_2 \in \Upsilon(\varphi_2) \\ \Upsilon(\varphi_1) \cup \Upsilon(\varphi_2) & \textit{otherwise} \end{cases}$$

$$\Upsilon(\varphi_1\, \boldsymbol{U} \varphi_2) = \Upsilon(\varphi_1) \cup \Upsilon(\varphi_2) \qquad \Upsilon(\ell) = \ell \quad \textit{for literals } \ell$$

For a formula $\varphi \in$ LTL\$GU$ *we define its* until factor *as the formula* $\varphi_{/\Upsilon(\varphi)} \in$ LTL(U) *simply written as* χ_φ.

The following proposition relates the semantics of an LTL\GU formula with the semantics of its until and until-free factors.

Proposition 19. *For any* $\varphi \in$ LTL\$GU$ *over propositions* P *and any relation* $R \subseteq (2^P)^\omega \times (2^{P_{\Upsilon(\varphi)}})^\omega$ *such that* $R \triangleleft R_{\Upsilon(\varphi)}$ *we have*

$$[\![\varphi]\!] = (R^\circ \cap [\![\chi_\varphi]\!]) \restriction_{2^P}$$

Proof. We prove our statement by proving the following equivalence

$$w \vDash \varphi \iff (w \cup \lambda(w)) \vDash \chi_\varphi \tag{1}$$

where $\lambda(w) \in (2^{P_{\Upsilon(\varphi)}})^\omega$ is the unique word such that $(w, \lambda(w)) \in R_{\Upsilon(\varphi)}$. We do so by performing induction on φ:

i. $\varphi \in \Upsilon(\varphi)$: in which case $\chi_\varphi = p_\varphi$.

$$\begin{aligned}
& w \vDash \varphi \\
\iff \quad & p_\varphi \in \lambda(w)[0] && \text{(definition of } \lambda(w)\text{)} \\
\iff \quad & (w \cup \lambda(w)) \vDash p_\varphi
\end{aligned}$$

ii. $\varphi = (\varphi_1 \wedge/\vee \varphi_2) \notin \Upsilon(\varphi_1 \wedge/\vee \varphi_2)$: here $\chi_{\varphi_1 \wedge/\vee \varphi_2} = \chi_{\varphi_1} \wedge/\vee \chi_{\varphi_2}$

$$\begin{aligned}
& w \vDash \varphi_1 \wedge/\vee \varphi_2 \\
\iff \quad & w \vDash \varphi_1 \quad \text{and/or} \quad w \vDash \varphi_2 \\
\iff \quad & (w \cup \lambda(w)) \vDash \chi_{\varphi_1} \quad \text{and/or} \quad (w \cup \lambda(w)) \vDash \chi_{\varphi_2} && \text{(inductive hypothesis)} \\
\iff \quad & (w \cup \lambda(w)) \vDash (\chi_{\varphi_1} \wedge/\vee \chi_{\varphi_1}) = \chi_{\varphi_1 \wedge/\vee \varphi_2}
\end{aligned}$$

iii. $\varphi = (\varphi_1 \boldsymbol{U} \varphi_2)$: here $\chi_{\varphi_1 \boldsymbol{U} \varphi_2} = \chi_{\varphi_1} \boldsymbol{U} \chi_{\varphi_2}$

$$\begin{aligned}
& w \vDash \varphi_1 \boldsymbol{U} \varphi_2 \\
\iff \quad & \exists\, i \;\; w_i \vDash \varphi_2 \quad \text{and} \quad \forall\, j{<}i \;\; w_j \vDash \varphi_1 \\
\iff \quad & \exists\, i \;\; (w_i \cup \lambda(w_i)) \vDash \chi_{\varphi_2} \quad \text{and} \quad \forall\, j{<}i \;\; (w_j \cup \lambda(w_j)) \vDash \chi_{\varphi_1} \\
& && \text{(inductive hypothesis)} \\
\iff \quad & (w \cup \lambda(w)) \vDash \chi_{\varphi_1} \boldsymbol{U} \chi_{\varphi_2} = \chi_{\varphi_1 \boldsymbol{U} \varphi_2}
\end{aligned}$$

Now we also have $(w \cup \lambda(w)) \in R^\circ$ due to the fact that $(w, \lambda(w)) \in R_{\Upsilon(\varphi)}$ and $R_{\Upsilon(\varphi)} \subseteq R$. This along with (1) gives us $[\![\varphi]\!] \subseteq (R^\circ \cap [\![\chi_\varphi]\!]) \restriction_{2^P}$.

Next we observe that if $\lambda_1 \subseteq \lambda_2$ then $w \cup \lambda_1 \vDash \chi_\varphi$ implies $w \cup \lambda_2 \vDash \chi_\varphi$ because the propositions in $P_{\Upsilon(\varphi)}$ appear positively in χ_φ. Consider $w \in (R^\circ \cap [\![\chi_\varphi]\!]) \restriction_{2^P}$, this implies there is a λ such that $(w, \lambda) \in R$ and $(w \cup \lambda) \vDash \chi_\varphi$. Since $R \lhd R_{\Upsilon(\varphi)}$ we have $\lambda \subseteq \lambda(w)$ where $(w, \lambda(w)) \in R_{\Upsilon(\varphi)} \subseteq R$. Now from our first observation in this paragraph we have that $(w \cup \lambda(w)) \vDash \chi_\varphi$, from which we can conclude $w \vDash \varphi$ using (1). This proves the other side of the containment $(R^\circ \cap [\![\chi_\varphi]\!]) \restriction_{2^P} \subseteq [\![\varphi]\!]$. □

Let P_1 and P_2 be disjoint set of propositions. Let $\Sigma = 2^{P_1}$, $\Gamma = 2^{P_2}$ and by abusing notation let $\Sigma \times \Gamma = 2^{P_1 \cup P_2}$. Next we describe how to compose a master NBA $\mathcal{A}$ over input $\Sigma \times \Gamma$ and a slave NBM $\mathcal{B}$ over input alphabet Σ and output alphabet Γ to obtain an NBA $\mathcal{A} \times \mathcal{B}$ over input Σ.

Definition 20 (Composition). *Consider a NBA $\mathcal{A} = (Q_\mathcal{A}, \delta_\mathcal{A}, I_\mathcal{A}, F_\mathcal{A})$ over input alphabet $\Sigma \times \Gamma$ and a NBM $\mathcal{B} = (Q_B, \delta_\mathcal{B}, I_\mathcal{B}, M_\mathcal{B}, F_\mathcal{B})$ over input alphabet Σ and output alphabet Γ. We define a NBA $(Q_\mathcal{A} \times Q_\mathcal{B}, \delta_{\mathcal{A}\times\mathcal{B}}, I_{\mathcal{A}\times\mathcal{B}}, F_\mathcal{A} \times F_\mathcal{B})$ over input Σ denoted by $\mathcal{A} \times \mathcal{B}$ where*

$$\delta_{\mathcal{A}\times\mathcal{B}}((a_1, b_1), \sigma) = \{(a_2, b_2) \mid q_2 \in \delta_\mathcal{B}(q_1, \sigma),\ a_2 \in \delta_\mathcal{A}(a_1, \sigma \cup M_\mathcal{B}(b_1, \sigma))\}$$

$$I_{\mathcal{A}\times\mathcal{B}} = \{(a_1, b_1) \xrightarrow{\nu} (a_2, b_2) \mid b_1 \xrightarrow{\nu} b_2 \in I_\mathcal{B},\ a_1 \xrightarrow{\nu \cup M_\mathcal{B}(b_1, \nu)} a_2 \in I_\mathcal{A}\}$$

The proposition below relates the languages accepted by a master and a slave NBA to the language accepted by the product defined above. It requires the master to have final states such that on entering a final state it can never leave the set of final states. We shall refer to this property as absorbing final states. We provide a complete proof of this result in the technical report [9].

Proposition 21. *Given a NBA $\mathcal{A}$ with input alphabet $\Sigma \times \Gamma$ with absorbing final states, and a NBA $\mathcal{B}$ with input alphabet Σ and output alphabet Γ we have*

$$L(\mathcal{A} \times \mathcal{B}) = (R_{\mathcal{B}}{}^{\circ} \cap L(\mathcal{A})) \restriction_{\Sigma}$$

The following proposition shows that the composition of a master and slave is limit deterministic if both of them are limit deterministic. The proof is provided in the technical report [9].

Proposition 22. *Given a limit deterministic NBA $\mathcal{A}$ over input alphabet $\Sigma \times \Gamma$, and a limit deterministic NBM $\mathcal{B}$ with input alphabet Σ and output alphabet Γ it is the case that $\mathcal{A} \times \mathcal{B}$ is also limit deterministic.*

The next proposition illustrates how to construct a Mealy machine which recognizes a relation which is subsumed by the characteristic relation of an until-free factor, thus constructing a slave automaton of exponential size.

Proposition 23. *For any finite set $\Phi \subset$ LTL(F,G) over propositions P there is a NBM $\mathcal{B}_\Phi$ with input over 2^P and output over 2^{P_Φ} such that $R_{\mathcal{B}_\Phi} \lhd R_\Phi$, $\mathcal{B}_\Phi$ is limit deterministic and of size $\mathcal{O}(2^{|\Phi|})$.*

Proof. Consider the construction of Theorem 15 with the following modifications to construct $\mathcal{B}_\Phi$:

- let Π be set of all three way partition of $\mathcal{G}_\Phi \cup \mathcal{F}_\Phi$ instead of $\mathcal{G}_\varphi \cup \mathcal{F}_\varphi$
- every transition of the form $(\pi, 0) \xrightarrow{\nu} (\pi', m)$ is initial
- define $M_{\mathcal{B}}((\pi, n), \nu)$ as $\{\, p_\varphi \in P_\Phi \mid [\![\, \varphi \,]\!]^\pi_\nu = \mathit{true} \,\}$

The proof of $R_{\mathcal{B}_\Phi} \lhd R_\Phi$ can be found in the technical report [9]. □

Next we observe that master automaton can be constructed using a standard approach of translating an alternating automaton for the until factor to an NBA.

Proposition 24. *[18] For any formula $\varphi \in$ LTL(U) there is a NBA $\mathcal{A}_\varphi$ with a single absorbing final state such that $L(\mathcal{A}_\varphi) = [\![\varphi]\!]$ and is of size $\mathcal{O}(2^{|\varphi|})$.*

Proof. As observed in Lemma 2 in [8], the construction follows from Theorem 22 and Proposition 20 of [18]. □

Finally we combine all the results in this section to show that the composition of the master and slave for the until and the until free components is as desired.

Theorem 25. *For any formula $\varphi \in \text{LTL}\backslash GU$ the NBA $\mathcal{A}_{\chi_\varphi} \times \mathcal{B}_{\Upsilon(\varphi)}$ recognizes $[\![\varphi]\!]$, is limit deterministic and is of size $2^{\mathcal{O}(|\varphi|)}$.*

Proof. First we look at the language recognized by $\mathcal{A}_{\chi_\varphi} \times \mathcal{B}_{\Upsilon(\varphi)}$:

$$
\begin{aligned}
& L(\mathcal{A}_{\chi_\varphi} \times \mathcal{B}_{\Upsilon(\varphi)}) \\
& = (R_{\mathcal{B}_{\Upsilon(\varphi)}}{}^{\circ} \cap L(\mathcal{A}_{\chi_\varphi})) \restriction_{2^P} && \text{(Proposition 21)} \\
& = (R_{\mathcal{B}_{\Upsilon(\varphi)}}{}^{\circ} \cap [\![\chi_\varphi]\!]) \restriction_{2^P} && \text{(Proposition 24)} \\
& = [\![\varphi]\!] && \text{(Proposition 19 \& 23)}
\end{aligned}
$$

We have $\mathcal{A}_{\chi_\varphi} \times \mathcal{B}_{\Upsilon(\varphi)}$ to be limit deterministic from Proposition 22. The automata $\mathcal{A}_{\chi_\varphi}$ and $\mathcal{B}_{\Upsilon(\varphi)}$ are both exponential in the size of φ as seen in Propositions 24 & 23, and so the product is also of size $2^{\mathcal{O}(|\varphi|)}$. □

5 Results and Applications

In this section, we summarize and reflect on some of the consequences of our results related to PBAs and model checking concurrent probabilistic programs.

5.1 Model Checking MDPs

MDPs are the prevalent models for concurrent probabilistic programs. Concurrency is modeled as nondeterministic states, where the transitions are chosen by a scheduler. We refer the reader to [5] for a complete definition of MDPs. The model checking problem can be formulated as follows.

Definition 26. *Given a MDP $\mathcal{N}$ and temporal logic formula φ, the model checking problem is to decide if there exists a scheduler u such that $\Pr_{\mathcal{N},u}([\![\varphi]\!]) > 0$.*

Our construction of limit deterministic automata for $\text{LTL}\backslash GU$ leads to an improved EXPTIME model checking algorithm for MDPs and formulae in this logic which is matched by an EXPTIME-hard lower bound.

Theorem 27. *The model checking problem for MDPs and formulae in* $\text{LTL}\backslash GU$ *is* EXPTIME*-complete.*

Proof. Proposition 4.2.3 in [5] states one can decide the model checking problem of MDPs and limit deterministic NBAs by taking a product of the two and doing a linear time analysis of the resulting graph. Using this along with our result in Theorem 25 we obtain the required upper bound. Proof of the lower bound can be found in the technical report [9]. □

We contrast this with the complexity of model checking MDPs and full LTL.

Proposition 28. *[5] The model checking problem for MDPs and formulae in* LTL *is* 2EXPTIME*-complete.*

5.2 PBAs for LTL

First, we observe that our construction of limit deterministic NBAs gives us an exponential upper-bound on the size of PBAs for LTL\GU.

Theorem 29. *For any formula $\varphi \in$ LTL\$GU$ there is a PBA $\mathcal{M}$ such that $L_{>0}(\mathcal{M}) = [\![\varphi]\!]$ and is of size $2^{\mathcal{O}(|\varphi|)}$.*

Proof. Follows from Lemma 8 and Theorem 25. □

Next, we note that this upper-bound is the best one can achieve.

Proposition 30. *There exists a family of formulae $\varphi_n \in \mathrm{LTL}(G)$ of size n such that any PBAs recognizing them have size at least 2^n.*

Proof. Consider the formula $\boldsymbol{G}(p \iff \boldsymbol{X}^n p)$. The language accepted by it is the set $\{u^\omega \mid u \in \Sigma^n\}$ where $\Sigma = \{\emptyset, \{p\}\}$. For each $u \in \Sigma^n$ there needs to be at least one state q_u such that u can reach q_u from the initial state with non-zero probability and no other $v \in \Sigma^n$ can reach q_u from the initial state with non-zero probability, because the word uv^ω should not be accepted with positive probability whereas u^ω should be accepted. □

Next, we note that constructing deterministic automata can result in a double exponential blow-up.

Proposition 31. *[1] There exists a family of of formulae $\varphi_n \in \mathrm{LTL}(F)$ such that any deterministic automata that recognize them have size $2^{2^{\Omega(n)}}$.*

6 Conclusions

In this paper, we have presented a translation of formulae in LTL\$GU$ to limit deterministic automata that are provably exponentially smaller than deterministic automata for this logic. This yields a new, improved exponential time algorithm for model checking MDPs and LTL\$GU$ as compared to the previously known double exponential time complexity for MDPs and full LTL. It also gives us a way to build PBAs of exponential size for LTL\$GU$. Our automata in addition to having better upper-bounds also have a well defined logical structure, which makes it amenable to several optimizations.

There are few questions that are still left open. While we have shown how to construct exponential sized probabilistic and limit deterministic automata for LTL\GU, it still remains open whether we can construct equally small probabilistic or limit deterministic automata for full LTL. In [8] we prove that translating the safety fragment of LTL to weak probabilistic monitors (which are special PBAs) can result in a double exponential blow-up. This might indicate that it is unlikely one will be able to give exponential sized PBAs for full LTL. While Proposition 28 excludes the possibility of constructing such automata in exponential time, it does not rule out the existence of exponential sized automata for full LTL which can potentially be built in double exponential time.

As a part of future work we intend to implement our construction and compare it with results for deterministic automata, and also see if our new algorithm for model checking yields better performance in practice.

Acknowledgments. Dileep Kini was partially supported by NSF grant CNS-1016791 and Mahesh Viswanathan by NSF grant CNS-1314485.

References

1. Alur, R., Torre, S.L.: Deterministic generators and games for ltl fragments. ACM Trans. Comput. Logic 5(1), 1–25 (2004)
2. Babiak, T., Blahoudek, F., Křetínský, M., Strejček, J.: Effective translation of LTL to deterministic Rabin automata: Beyond the (F,G)-fragment. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 24–39. Springer, Heidelberg (2013)
3. Baier, C., Größer, M.: Recognizing omega-regular languages with probabilistic automata. In: LICS, pp. 137–146 (2005)
4. Chadha, R., Sistla, A.P., Viswanathan, M.: On the expressiveness and complexity of randomization in finite state monitors. J. ACM 56(5), 26:1–26:44 (2009)
5. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM 42(4), 857–907 (1995)
6. Esparza, J., Křetínský, J.: From LTL to deterministic automata: A safraless compositional approach. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 192–208. Springer, Heidelberg (2014)
7. Gastin, P., Oddoux, D.: Fast LTL to büchi automata translation. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 53–65. Springer, Heidelberg (2001)
8. Kini, D., Viswanathan, M.: Probabilistic automata for safety LTL specifications. In: McMillan, K.L., Rival, X. (eds.) VMCAI 2014. LNCS, vol. 8318, pp. 118–136. Springer, Heidelberg (2014)
9. Kini, D., Viswanathan, M.: Probabilistic büchi automata for LTL\GU. Technical Report University of Illinois at Urbana-Champaign (2015), `http://hdl.handle.net/2142/72686`
10. Klein, J., Baier, C.: Experiments with deterministic ω-automata for formulas of linear temporal logic. Theoretical Computer Science 363(2), 182–195 (2006)
11. Křetínský, J., Esparza, J.: Deterministic automata for the (F,G)-fragment of LTL. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 7–22. Springer, Heidelberg (2012)
12. Křetínský, J., Garza, R.L.: Rabinizer 2: Small deterministic automata for LTL\ GU. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 446–450. Springer, Heidelberg (2013)
13. Morgenstern, A., Schneider, K.: From LTL to symbolically represented deterministic automata. In: Logozzo, F., Peled, D.A., Zuck, L.D. (eds.) VMCAI 2008. LNCS, vol. 4905, pp. 279–293. Springer, Heidelberg (2008)
14. Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 364–380. Springer, Heidelberg (2006)
15. Pnueli, A., Zaks, A.: On the merits of temporal testers. In: 25 Years of Model Checking, pp. 172–195 (2008)
16. Somenzi, F., Bloem, R.: Efficient Büchi automata from LTL formulae. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 248–263. Springer, Heidelberg (2000)
17. Vardi, M., Wolper, P., Sistla, A.P.: Reasoning about infinite computation paths. In: FOCS (1983)
18. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency. LNCS, vol. 1043, pp. 238–266. Springer, Heidelberg (1996)