Optimal Translation of LTL to Limit Deterministic Automata

Dileep Kini* and Mahesh Viswanathan*

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

Abstract. A crucial step in model checking Markov Decision Processes (MDP) is to translate the LTL specification into automata. Efforts have been made in improving deterministic automata construction for LTL but such translations are double exponential in the worst case. For model checking MDPs though limit deterministic automata suffice. Recently it was shown how to translate the fragment LTL$\setminus GU$ to exponential sized limit deterministic automata which speeds up the model checking problem by an exponential factor for that fragment. In this paper we show how to construct limit deterministic automata for full LTL. This translation is not only efficient for LTL$\setminus GU$ but for a larger fragment LTL_{D} which is provably more expressive. We show experimental results demonstrating that our construction yields smaller automata when compared to state of the art techniques that translate LTL to deterministic and limit deterministic automata.

1 Introduction

Markov Decision Processes (MDPs) [20, 24, 4] are the canonical model used to define the semantics of systems like concurrently running probabilistic programs that exhibit both stochastic and nondeterministic behavior. MDPs are interpreted with respect to a scheduler that resolves the nondeterminism. Such a scheduler chooses a probabilistic transition from a state based on the past sequence of states visited during the computation. When undesirable system behaviors are described by a formula φ in linear temporal logic (LTL), *qualitative verification* involves checking if there is some (adversarial) scheduler with respect to which the measure of paths satisfying φ is non-zero. Model checking algorithms [4] in this context proceed by translating the LTL requirement φ into an automaton $\mathcal{A}$, taking the synchronous cross-product of the MDP model M and the automaton $\mathcal{A}$ to construct a new MDP M', and finally, analyzing the MDP M' to check the desired property. The complexity of this procedure is polynomial in the size of the final MDP M', and hence critically depends on the size of automaton $\mathcal{A}$ that results from translating the LTL specification.

MDP model checking algorithms based on the above idea require the translated automaton to be of a special form as general non-deterministic automata

* Authors were supported by NSF grants CNS 1314485 and CCF 1422798.

are not sufficient. The Büchi automaton has to be either deterministic or *deterministic in the limit* — a Büchi automaton is deterministic in the limit if every state reachable from an accepting state has deterministic transitions[1]. Limit-determinism is also sometimes referred to as semi-determinism. Deterministic or limit deterministic automata for LTL formulae can be constructed by first translating the formula into a nondeterministic Büchi automaton, and then either determinizing or "limit-determinizing" the machine. This results in an automaton that is doubly exponential in the size of the LTL formula, which gives a 2EXPTIME algorithm for model checking MDPs.

Direct translations of LTL (and fragments of LTL) to deterministic Rabin automata have been proposed [11, 18, 17, 14, 3, 5]. However, any such translation, in the worst case, results in automata that are doubly exponential in size [2]; this holds for any fragment of LTL that contains the operators $\vee$, $\wedge$, and $\mathbf{F}$. Recently [9] a fragment of LTL called LTL$\setminus GU$ [15] was translated into limit deterministic Büchi automata. LTL$\setminus GU$ is a fragment of LTL where formulae are built from propositions and their negations using conjunction, disjunction, and the temporal operators $\mathbf{X}$ (next), $\mathbf{F}$ (eventually/finally), $\mathbf{G}$ (always/globally), and $\mathbf{U}$ (until), with the restriction that no $\mathbf{U}$ operator appears within the scope of a $\mathbf{G}$ operator. The most important feature of this translation from LTL$\setminus GU$ to limit deterministic automata is the fact that the resulting automaton is only exponential in the size of the formula. Thus, this automata construction can be used to obtain an EXPTIME algorithm for model checking MDP against LTL$\setminus GU$ formulas, as opposed to 2EXPTIME.

Recently, a translation from full LTL logic to limit deterministic automata has been proposed [21]. This translation is very similar to the translation to deterministic automata proposed in [5], with the use of nondeterminism being limited to simplifying the acceptance condition. Therefore, like the deterministic translations of LTL, it can be shown to construct doubly exponential sized automata even for very simple LTL fragments like those that contain $\vee$, $\wedge$, and $\mathbf{F}$. Thus, it does not achieve the optimal bounds for LTL$\setminus GU$ shown in [9]. However, one advantage of the construction in [21] is that it can be used in quantitative verification as well as qualitative verification of MDPs and has been implemented in [22]. Quantitative verification of MDPs can also be performed using nondeterministic automata that have the *good-for-games* (GFG) property [8, 12], but translating a general NBA into a GFG automaton is known to result in an exponential blow-up. An alternate approach to quantitative verification using subset/breakpoint construction on a NBA is proposed in [7] but it also suffers from an exponential blow up.

In this paper we continue the line of work started in [9, 21], and present a new translation of the full LTL logic to limit deterministic Büchi automata. The new

[1] Limit deterministic automata are not the same as *unambiguous automata*. Unambiguous automata have at most one accepting run for any input. It is well known that every LTL formula can be translated into an unambiguous automaton of exponential size [23]. This has been shown to be not true for limit deterministic automata in [21].

translation can be shown to be a generalization of the construction in [9] in that it constructs exponential sized automata for LTL$\backslash GU$. In fact, we show that this new translation yields exponential sized automata for a richer fragment of LTL that we call LTL_D (see Section 5 for a comparison between the expressive powers of LTL_D and LTL$\backslash GU$). This improves the complexity of qualitative MDP model checking against LTL_D to EXPTIME from 2EXPTIME.

Our automaton construction uses two main ideas. The first is an idea discovered in [9]. To achieve limit determinism, for certain subformulae ψ of φ, the automaton of φ tracks how often $\mathbf{F}\psi$ and $\mathbf{G}\psi$ formulae are true; this is in addition to tracking the truth (implicitly) of all subformulae ψ, as all translations from LTL to automata do. Second, for untils within the scope of $\mathbf{G}$, we do a form of subset construction that ensures that the state explores all the possible ways in which such formulae can be satisfied in the future, and for untils outside the scope of $\mathbf{G}$ we use non-determinism to check its truth.

We have implemented our translation from LTL to limit deterministic automata in a tool called `Büchifier`. We show experimental results demonstrating that in most cases our construction yields smaller automata when compared to state of the art techniques that translate LTL to deterministic and limit deterministic automata.

2 Preliminaries

First we introduce the notation we use throughout the paper. We use P to denote the set of propositions. We use w to denote infinite words over a finite alphabet. We use w_i to denote the i^{th} (index starting at 0) symbol in the sequence w, and use $w[i]$ to denote the suffix $w_i w_{i+1} \dots$ of w starting at i. We use $w[i,j]$ to denote the substring $w_i \dots w_{j-1}$. We use $[n]$ to denote all non-negative integers less than n that is $\{0, 1, \dots, n-1\}$. We begin by recalling the syntax of LTL:

Definition 1 (LTL Syntax). *Formulae in* LTL *are given by the following syntax:*

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

Next, we look at the semantics of the various operators:

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

$$\begin{aligned}
w \vDash p\ (\neg p) &\iff p \in w_0\ (p \notin w_0) & \qquad w \vDash \mathbf{X}\varphi &\iff w[1] \vDash \varphi \\
w \vDash \varphi \vee \psi &\iff w \vDash \varphi \text{ or } w \vDash \psi & \qquad w \vDash \mathbf{F}\varphi &\iff \exists\, i : w[i] \vDash \varphi \\
w \vDash \varphi \wedge \psi &\iff w \vDash \varphi \text{ and } w \vDash \psi & \qquad w \vDash \mathbf{G}\varphi &\iff \forall\, i : w[i] \vDash \varphi \\
w \vDash \varphi \,\mathbf{U}\, \psi &\iff \exists\, i : w[i] \vDash \psi, \text{and} & & \\
& \qquad \forall\, j < i : w[j] \vDash \varphi & &
\end{aligned}$$

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

(Note that the release operator $\mathbf{R}$, the dual of $\mathbf{U}$, can be expressed using $\mathbf{U}$ and $\mathbf{G}$, i.e $\psi_1 \mathbf{R} \psi_2 \equiv (\psi_2 \mathbf{U} (\psi_1 \wedge \psi_2)) \vee \mathbf{G}\psi_2$. Hence we omit it from any of the logics we consider.)

In this paper the terminology *subformula of* φ is used to denote a node within the parse tree of φ. When we refer to the subformula as an LTL formula we will be referring to the formula at that node. Two subformulae that have the same formulae at their nodes need not be the same owing to the possibility of them being in different contexts. This distinction will be important as we treat formulae differently depending on their contexts. For the purposes of describing different subfragments we qualify subformulae as being either *internal* or *external*.

Definition 3. *A subformula ψ of φ is said to be* internal *if ψ is in the scope of some $\mathbf{G}$-subformula of φ, otherwise it is said to be* external.

Many syntactic restrictions of LTL have been considered for the sake of obtaining smaller automata translations. LTL(F,G) (read "LTL F G") and LTL$\backslash GU$ (read "LTL set minus G U") are two such fragments which we recall in the next two definitions.

Definition 4 (LTL(F,G) Syntax). *The fragment* LTL(F,G) *over propositions P is described by the following syntax*

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

Definition 5 (LTL\GU Syntax). *The fragment* LTL$\backslash GU$ *is given by the syntax*

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

LTL(F,G) allows for $\mathbf{G}$ and $\mathbf{F}$ as the only temporal operators. The fragment LTL$\backslash GU$ additionally allows for external $\mathbf{U}$ but not internal ones. Also, we choose to represent an external $\mathbf{F}$ using $\mathbf{U}$. In other words every $\mathbf{F}$ will be internal. Next, we introduce the fragment LTL_D (read "LTL D")

Definition 6 (LTL_D Syntax). *The formulae in the fragment* LTL_D *are given by the syntax for ϑ:*

$$\begin{aligned} \psi \quad &::= \quad \varphi \mid \psi \vee \varphi \mid \varphi \vee \psi \mid \psi \wedge \psi \mid \psi \mathbf{U} \varphi \mid \mathbf{G}\psi \mid \mathbf{X}\psi \qquad \varphi \in \text{LTL}(F,G) \\ \vartheta \quad &::= \quad \psi \mid \vartheta \vee \vartheta \mid \vartheta \wedge \vartheta \mid \vartheta \mathbf{U} \vartheta \mid \mathbf{X}\vartheta \end{aligned}$$

Unlike LTL$\backslash GU$, LTL_D allows for internal $\mathbf{U}$ but it is restricted. The following restrictions apply on LTL_D:

1. The second argument of every internal $\mathbf{U}$ formula is in LTL(F,G)
2. At least one argument of every internal $\vee$ is in LTL(F,G)

Note that LTL_D is strictly larger than LTL$\backslash GU$ in the syntactic sense, as every LTL$\backslash GU$ formula is also an LTL_D formula. We shall show in Section 5 that it is strictly richer in the semantic sense as well.

Next we define depth and height. A subformula ψ of φ is said to be at depth k if the number of $\mathbf{X}$ operators in φ within which ψ appears is exactly k. The height of a formula is the maximum depth of any of its subformulae.

Definition 7 (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; $I \subseteq Q$ is a set of initial states and $F \subseteq Q$ is a set of final states.*

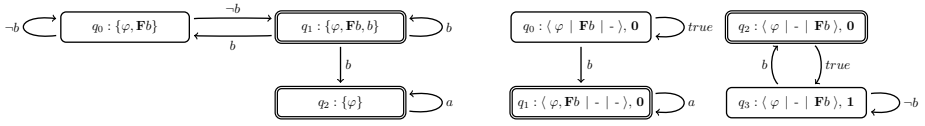
A run of a word $w \in \Sigma^\omega$ over a NBA is an infinite sequence of states $q_0 q_1 q_2 \ldots$ such that $q_0 \in I$ and $\forall\, i \geq 0$ $(q_i, w_i, q_{i+1}) \in \delta$. A run is accepting if $q_i \in F$ for infinitely many i.

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 8 (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 $\sigma \in \Sigma$.*

3 Construction

In this section we show our construction of limit deterministic automata for full LTL. First, let us look at an example that shows that the standard construction (Fischer-Ladner and its variants) is not limit deterministic. The standard construction involves guessing the set of subformulae that are true at each step and ensuring the guess is correct. For $\varphi = \mathbf{G}(a \vee \mathbf{F}b)$ this gives us the automaton (after pruning unreachable states and merging bisimilar ones. Here all 3 states are initial) in Figure 1a which is not limit deterministic as the final state q_1 has non-deterministic choices enabled.



(a) Standard Construction (b) Tripartition Construction

Fig. 1: Automata for $\mathbf{G}(a \vee \mathbf{F}b)$

Our construction builds upon the idea introduced in [9] of keeping track of how often **F**,**G**-subformulae are true. Therefore, we will incrementally describe the features of our automaton: first by revisiting the technique required for LTL(F,G) without **X**s, later by introducing the new ideas required to handle the untils and nexts.

Given an LTL(F,G) formula, for each of its **G**-subformula we are going to predict whether it is: always true (α), true at some point but not always (β), never true (γ). Note that for any formula if we predict α/γ then the prediction should remain the same going forward. For a **G**-subformula, $\mathbf{G}\psi$, if we predict β it means we are asserting $\mathbf{FG}\psi \wedge \neg\mathbf{G}\psi$ and therefore the prediction should

remain β until a certain point and then change to α. This prediction entails two kinds of non-deterministic choices: **(i)** the initial choice of assigning one of α, β, γ **(ii)** if assigned β initially then the choice of the time point at which to change it to α. The first choice needs to be made once at the beginning and the second choice has to be made eventually in a finite time. They together only constitute finitely many choices which is the source of the limit determinism. We similarly define predictions for **F**-subformulae as: never true (α), true at some point but not always (β), always true (γ). We flip the meaning of α and γ to ensure β becomes α eventually as for **G**-subformulae. An *FG-prediction* for a formula $\varphi \in$ LTL(F,G), denoted by π, is a tri-partition $\langle \alpha(\pi), \beta(\pi), \gamma(\pi) \rangle$ of its $\mathbf{F}, \mathbf{G}$-subformulae. We drop π when it is clear from the context. The *prediction* for a subformula ψ made by π is said to be $\alpha/\beta/\gamma$ depending upon the partition of π in which ψ is present. The space of all FG-predictions for φ is denoted by $\Pi(\varphi)$.

Example 1. Consider the formula $\varphi = \mathbf{G}(a \vee \mathbf{F}b)$, and an FG-prediction $\pi = \langle \alpha, \beta, \gamma \rangle$ for φ where $\alpha = \{\varphi\}$, $\beta = \{\mathbf{F}b\}$ and $\gamma = \emptyset$. For the formula φ the prediction made is α. Since it is a **G**-formula this prediction says that φ is always true or simply φ is true. For the subformula $\mathbf{F}b$ the prediction made is β. This prediction says that $\mathbf{F}b$ is true at some point but not always which implies $\mathbf{F}b$ is true but not $\mathbf{GF}b$.

The automaton for LTL(F,G) essentially makes a non-deterministic choice for π initially and at each step makes a choice of whether to move some formula(e) from β to α. The correctness of predictions made by π is monitored inductively. Suppose our prediction for a formula $\mathbf{G}\psi$ is α at some instant: this implies we need to check that ψ is true at every time point there onwards (or equivalently check that ψ is true whenever α is predicted for $\mathbf{G}\psi$ since the prediction α never changes). If we are able to monitor the ***truth*** of ψ at every instant then it is clear how this can be used to monitor the ***prediction*** α for $\mathbf{G}\psi$. The crucial observation here is that the correct prediction for $\mathbf{G}/\mathbf{F}$ formula gives us their truth: a $\mathbf{G}/\mathbf{F}$ formula is true/false (respectively) at a time point if and only if its correct prediction is α at that time. Now the prediction α for $\mathbf{G}\psi$ can be checked by using the truths (derived from the predictions) of the subformulae of ψ (inductive step). If ψ is propositional then its truth is readily available from the input symbol being seen (base case of the induction). This inductive idea shall be used for all predictions. Note that since our formulae are in negation normal form we only need to verify a prediction is correct if it asserts the truth rather than falsehood of a subformula. Therefore the predictions β, γ for $\mathbf{G}\psi$ need not be checked. In case of $\mathbf{F}\psi$ the prediction α need not be checked (as it entails falsehood of $\mathbf{F}\psi$) but β, γ do need to be checked. If our prediction for $\mathbf{F}\psi$ is β then we are asserting ψ is true until a certain point in the future at which the prediction becomes α. Therefore we only need to check that ψ is true when the prediction for $\mathbf{F}\psi$ changes to α. Once again we can inductively obtain the truth of ψ at that instant from the predictions for the subformulae of ψ and from the next input. For checking a prediction γ about $\mathbf{F}\psi$ we need to check ψ is true

infinitely often. For this purpose we use the Büchi acceptance where the final states are those where ψ is evaluated to be true, again inductively. When we are monitoring multiple $\mathbf{F}\psi$ for γ we will need a counter to cycle through all the $\mathbf{F}\psi$ in γ. Let m be the number of $\mathbf{F}\psi$ in γ. Observe that the set of formulae predicted to be γ never changes once fixed at the beginning and hence m is well defined. When the counter has value n, it is incremented cyclically to $n+1 (\text{mod } m)$ whenever the ψ corresponding to the n^{th} $\mathbf{F}\psi \in \gamma$ evaluates to true. The initial states are those in which the top formula evaluates to true given the predictions in that state. The final states are those where no formula is assigned β and the counter is 0. Summarizing, a state in our automata has two components: **(a)** an FG-prediction $\pi = \langle \alpha, \beta, \gamma \rangle$ (a tri-partition of the $\mathbf{F}, \mathbf{G}$-subformulae) and **(b)** a cyclic integer counter n. The transitions are determined by how the predictions and counters are allowed to change as described. We illustrate the construction using once again the formula $\varphi = \mathbf{G}(a \vee \mathbf{F}b)$ for which the automaton is presented in Figure 1b and its details are completely described in Appendix A.

3.1 Handling Untils and Nexts

Next we observe that the above technique does not lend itself to the $\mathbf{U}/\mathbf{X}$ operators. The crucial property used above about $\mathbf{F}, \mathbf{G}$-formulae is that they cannot be simultaneously infinitely often true and infinitely often false unlike $\mathbf{U}/\mathbf{X}$ formulae. So if we tried the above technique for $\mathbf{U}/\mathbf{X}$ we would not get limit determinism since the truth of the $\mathbf{U}/\mathbf{X}$ formulae would have to be guessed infinitely often.

The key idea we use in handling $\mathbf{U}/\mathbf{X}$s is to propagate their *obligation* along the states. Let us say the automaton needs to check if a formula φ holds for an input w, and it begins by making an FG-prediction π about w. The obligation when no input has been seen is φ. When the first symbol w_0 is seen it needs to update the obligation to reflect what "remains to be checked" for the rest of the input $w[1]$, in order for $w \vDash \varphi$ to hold, assuming π is correct for w. The automaton can keep updating the obligation as it sees each input symbol. The claim will be that the obligation is never falsified iff $w \vDash \varphi$, given that π is correct. This brings up some questions:

1. How are we exploiting opportunities for non-determinism?
2. How is the obligation computed at each step?
3. How is π checked to be correct in the presence of $\mathbf{U}/\mathbf{X}$s?

Exploiting non-determinism. Being able to exploit non-determinism helps in reducing the size of the automaton we construct. So the question is: how are we exploiting any opportunities for non-determinism (albeit for finite time)? The answer is to update the obligation non-deterministically. Checking the formula $\psi_1 \mathbf{U} \psi_2$ presents us with two alternatives: either ψ_2 is true now or $\psi_1 \wedge \mathbf{X}(\psi_1 \mathbf{U} \psi_2)$ is true now. Similarly $\psi_1 \vee \psi_2$ brings up two alternatives. We can pick between the obligations of these two choices non-deterministically. But we should somehow ensure that we are only allowed to use this non-determinism finitely often. This is

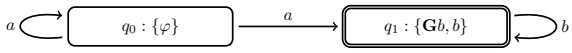


Fig. 2: Standard NBA construction for $\varphi = a\mathbf{U}(\mathbf{G}b)$.

where we treat internal and external (Definition 3) $\mathbf{U}/\vee$ subformulae differently. The observation is that external $\mathbf{U}/\vee$ need to be checked for only a finite amount of time. Hence the disjunctive choice presented by them can be dispatched non-deterministically each time without worrying about violating limit determinism. To illustrate this point we show the standard NBA for the formula $\varphi = a\mathbf{U}(\mathbf{G}b)$ in Figure 2 which turns out to be limit deterministic owing to the fact that the $\mathbf{U}$ is external. In Figure 1a we saw that the standard construction for $\varphi = \mathbf{G}(a \vee \mathbf{F}b)$ resulted in a NBA that was not limit-deterministic, and one of the reasons is that the $\mathbf{F}$, which is a special form of $\mathbf{U}$, is internal. An internal $\mathbf{U}/\vee$ may need to be checked infinitely many times and hence the choice should not be resolved non-deterministically, but carried forward as a disjunction of the obligations of the choices. Passing the choice forward without resolving it comes at a cost of a bigger state space, this is akin to the subset construction where all the choices are being kept track of.

Now we begin to formalize the ideas. To exploit the non-determinism allowed by the external $\mathbf{U}/\vee$ we introduce the concept of *ex-choice*. We use Λ_φ to denote the set of all external $\mathbf{U}/\vee$ subformulae. Any subset of it $\lambda \subseteq \Lambda_\varphi$ is called an ex-choice. An ex-choice dictates how each external $\mathbf{U}/\vee$ should be satisfied if it needs to be satisfied. The interpretation associated with λ is the following: if $\psi_1 \mathbf{U} \psi_2 \in \lambda$ then ψ_2 has to hold or if $\psi_1 \mathbf{U} \psi_2 \in \Lambda_\varphi - \lambda$ then $\psi_1 \wedge \mathbf{X}(\psi_1 \mathbf{U} \psi_2)$ has to hold. Similarly if $\psi_1 \vee \psi_2 \in \lambda$ then ψ_1 has to hold and if $\psi_1 \vee \psi_2 \in \Lambda_\varphi - \lambda$ then ψ_2 has to hold. The automaton we are going to construct is going to non-deterministically pick an ex-choice at each step and use it resolve the choices on external $\mathbf{U}/\vee$. After a finite time the ex-choice will not matter as the obligations will not consist of any external $\mathbf{U}/\vee$ that need to be checked (which will be enforced as a part of the acceptance condition), and hence limit determinism is ensured. The ex-choice picked along a transition is going to determine the obligation computed. Which leads us to the question of how the obligation is computed.

Computing Obligation. We define the *derivative* of a formula μ w.r.t an input symbol σ, FG-prediction π and ex-choice λ. The derivative should capture the obligation/requirement on any word ρ such that those obligations are able to imply that $\sigma\rho$ satisfies μ. This enables us to keep passing on the obligation forward as we see each symbol of the input by taking the derivative of the obligation so far. First, we need to ensure that the ex-choice λ picked when we are taking the transition dictates how a formula in Λ_φ should be satisfied if it needs to be. With that in mind we define $f(\lambda)$ as follows:

$$f(\lambda) = (\wedge_{(\phi \mathbf{U} \psi \in \lambda)} \phi \,\mathbf{U}\, \psi \Rightarrow \psi) \wedge (\wedge_{(\phi \mathbf{U} \psi \in (\Lambda_\varphi - \lambda))} \phi \,\mathbf{U}\, \psi \Rightarrow (\phi \wedge \mathbf{X}(\phi \,\mathbf{U}\, \psi))) \\ \wedge (\wedge_{(\phi \vee \psi \in \lambda)} \phi \vee \psi \Rightarrow \phi) \wedge (\wedge_{(\phi \vee \psi \in \Lambda_\varphi - \lambda)} \phi \vee \psi \Rightarrow \psi)$$

Since predictions made by π already tell us the truth of some of the subformulae, they need to be taken into account. Towards that we define the *substitution* of a formula ϕ w.r.t π, denoted by $[\,\phi\,]_\pi$ as the formula obtained from ϕ by substituting occurrences $\mathbf{G}\psi$ with $\mathbf{tt}$ if $\mathbf{G}\psi \in \alpha$ and $\mathbf{ff}$ otherwise, and similarly for $\mathbf{F}\psi$ with $\mathbf{ff}$ if $\mathbf{F}\psi \in \alpha$ and $\mathbf{tt}$ otherwise. The substitutions are done only for the maximal formulae in π that appear in ϕ, i.e., if ψ_1, ψ_2 are formulae in π such that ψ_1 is a subformula of ψ_2 then the substitution is not performed for ψ_1. Now we are ready to give a declarative definition of the derivative:

Definition 9. *Given an LTL formula μ over P, and a triple $\varepsilon = (\sigma, \pi, \lambda)$ where $\sigma \in 2^P$, $\pi \in \Pi(\varphi)$ and $\lambda \subseteq \Lambda_\varphi$: an LTL formula ψ is said to be a **derivative** of μ w.r.t to ε if*

$$\forall\, \rho \in \left(2^P\right)^\omega \quad \rho \vDash \psi \implies \sigma\rho \vDash [\,\mu \wedge f(\lambda)\,]_\pi$$

*The **weakest derivative** of μ w.r.t ε, denoted by $\nabla(\mu, \varepsilon)$, is a derivative such that $\psi \implies \nabla(\mu, \varepsilon)$ for any other derivative ψ.*

Since we will only be interested in the weakest derivative (as opposed to any other derivative) we shall refer to it as the derivative. The above definition is only declarative in the sense that it does not give us an explicit way to compute the derivative. We present this definition here for the sake of simplicity and ease of understanding for the reader. In the Appendix we provide a syntactic definition (Definition 15) and all the necessary machinery that allows us to compute such a formula. The syntactic definition also restricts the representation of the obligations to $\mathcal{B}^+(\varphi)$ which is the set of all positive Boolean combinations of subformulae of φ.

The automaton now will have an extra component μ corresponding to the obligation along with (π, n) from before. In the initial state μ will be the given formula φ that needs to be checked. At each step, the automaton sees an input symbol σ and makes a non-deterministic ex-choice $\lambda \subseteq \Lambda_\varphi$. The obligation at the next state will then become $\nabla(\mu, \varepsilon)$ where $\varepsilon = (\sigma, \pi, \lambda)$. The process continues as long as the obligation is never falsified. In order to ensure that every external until is dispatched in finite time, we impose that the obligation μ in the final states is *ex-free*, i.e. free of any formulae in Λ_φ. When the obligation is ex-free the ex-choice does not play a role in determining its derivative and we shall drop λ whenever that is the case, and this eliminates any non-determinism once a final state is visited. In order to ensure that an internal until, say $\phi\,\mathbf{U}\,\psi$ is not delayed forever, we involve $\mathbf{F}\psi$ in the FG-prediction and enhance the definition of substitution to say that $\phi\,\mathbf{U}\,\psi$ is replaced with $\mathbf{ff}$ if $\mathbf{F}\psi \in \alpha$. This way the derivative will impose that $\mathbf{F}\psi$ is true whenever $\phi\,\mathbf{U}\,\psi$ is claimed to be true. With this in mind we define the closure of φ, denoted by $\mathcal{C}(\varphi)$, to be set of all $\mathbf{F}, \mathbf{G}$-subformulae of φ, along with all $\mathbf{F}\psi$ for every internal $\phi\mathbf{U}\psi$ subformula of φ. We re-define an FG-prediction π to be any tri-partition of $\mathcal{C}(\varphi)$. Note that for every $\mathbf{F}\psi$ or $\mathbf{G}\psi$ in $\mathcal{C}(\varphi)$, ψ is internal.

Example 2. Let $\varphi = \mathbf{G}(\mathbf{F}a \vee (b\,\mathbf{U}\,c))$. Here $\mathcal{C}(\varphi) = \{\varphi, \mathbf{F}a, \mathbf{F}c\}$

Example 3. Let $\varphi = a\mathbf{U}(b \wedge \mathbf{G}c)$ be an internal subformula of some given formula. $\nabla(\varphi, \varepsilon)$ can take different values depending upon $\varepsilon = (\sigma, \pi)$. Here ex-choice λ does not play a role because the only $\mathbf{U}$ is internal. Note that $\varphi' = \mathbf{F}(b \wedge \mathbf{G}c)$ is in the closure. If $\varphi' \in \alpha$, then $\nabla(\varphi, \varepsilon) = \mathbf{ff}$ because $[\![\varphi]\!]_\pi$ would be $\mathbf{ff}$ owing to φ being substituted with $\mathbf{ff}$. Let $\varphi' \notin \alpha$. Now if $\mathbf{G}c \in \alpha$ then substituting $\mathbf{tt}$ in place of $\mathbf{G}c$ gives us $a\mathbf{U}b$ whose satisfaction depends upon the truth of a and b as given by σ. So if $\sigma(b) = \mathbf{tt}$ then the $\mathbf{U}$ is immediately satisfied and so $\nabla(\varphi, \varepsilon) = \mathbf{tt}$. If $\sigma(b) = \mathbf{ff}$ then the $\mathbf{U}$ is delayed and hence $\nabla(\varphi, \varepsilon)$ is either $a\mathbf{U}b$ or $\mathbf{ff}$ depending on $\sigma(a) = \mathbf{tt}/\mathbf{ff}$ respectively. If $\mathbf{G}c \notin \alpha$ then truth of b does not matter (as replacing $\mathbf{G}c$ with $\mathbf{ff}$ makes $b \wedge \mathbf{G}c = \mathbf{ff}$) and once again the derivative is $\varphi/\mathbf{ff}$ depending upon $\sigma(a)$.

Checking FG-predictions in the presence of untils and nexts. The main idea in being able to check an FG-prediction π was that a correct prediction about an $\mathbf{F}, \mathbf{G}$-subformula also tells us its truth. When we have $\mathbf{U}/\mathbf{X}$s in the mix, we no longer have a prediction available for them, and hence no immediate way to check if some subformula is true. For example when $\mathbf{G}\psi \in \alpha$ we needed to check ψ is true and we did so inductively using the predictions for subformulae in ψ. Now, since ψ can have $\mathbf{U}/\mathbf{X}$ within them it is not clear how we are going to check truth of ψ. In this case we pass ψ to the obligation μ. Similarly when the prediction of $\mathbf{F}\psi$ is changed from β to α we need to check ψ is true so once again we pass ψ to the obligation. So given consecutive FG-predictions π, π' define Ψ as the set

$$\Psi = \{\psi \mid \mathbf{F}\psi \in \beta(\pi) \cap \alpha(\pi') \text{ or } \mathbf{G}\psi \in \alpha(\pi)\} \tag{1}$$

and update the obligation along a transition $(\mu, \pi, n) \xrightarrow{\sigma} (\mu', \pi', n')$ as: $\mu' = \nabla(\mu \wedge (\wedge_{\psi \in \Psi} \psi), \varepsilon)$ where $\varepsilon = (\sigma, \pi, \lambda)$. Now consider the case when the counter is $n > 0$ and need to verify that the n^{th} $\mathbf{F}\psi$ formula in γ is true. In this case we cannot pass on ψ to the obligation because $\mathbf{F}\psi$ may be true because ψ is true at a later point and not now. Since we cannot predict when ψ is going to be true we carry the disjunction of all the derivatives of ψ since the counter was incremented to n. We keep doing it until this "carry" becomes true indicating that ψ became true at some point since we started checking for it. We also increment the counter at that point. This "carry" becomes yet another component ν in the automaton's state. We use $\mathbb{F}(S)$ to denote all $\mathbf{F}\psi$ in set S. Now we are ready to put the pieces together to formally describe the entire construction.

Definition 10 (Construction). *Given a formula $\varphi \in$ LTL over propositions P, let $\mathcal{D}(\varphi)$ be the NBA (Q, δ, I, F) over the alphabet 2^P defined as follows:*

- *Q is the set $\mathcal{B}^+(\varphi) \times \mathcal{B}^+(\varphi) \times \Pi(\varphi) \times [n]$ where $n = |\mathbb{F}(\mathcal{C}(\varphi))| + 1$*
- *δ is the set of all transitions $(\mu, \nu, \pi, m) \xrightarrow{\sigma} (\mu', \nu', \pi', m')$ such that*

  (a) *$\alpha(\pi) \subseteq \alpha(\pi')$ and $\gamma(\pi) = \gamma(\pi')$*

  (b) *$\mu' = \nabla(\mu \wedge \theta, \varepsilon)$ for some $\lambda \subseteq \Lambda_\varphi$*
  where $\theta = (\wedge_{\psi \in \Psi} \psi)$, Ψ as defined in (1) and $\varepsilon = (\sigma, \pi, \lambda)$

(c) $m' = \begin{cases} (m+1) \,(\mathrm{mod}\, |\mathbb{F}(\gamma)| + 1) & \nu = \mathbf{tt} \\ m & \text{otherwise} \end{cases}$

(d) $\nu' = \begin{cases} \psi_{m'} & \nu = \mathbf{tt} \\ \nabla(\nu, \varepsilon) \vee \psi_m & \text{otherwise} \end{cases}$

where $\{\mathbf{F}\psi_1, .., \mathbf{F}\psi_k\}$ *is an enumeration of* $\mathbb{F}(\gamma)$, $\psi_0 = \mathbf{tt}$ *and* $\varepsilon = (\sigma, \pi)$

- I *is all states of the form* $(\varphi, \mathbf{tt}, \pi, 0)$
- F *is all states of the form* $(\mu, \mathbf{tt}, \pi, 0)$ *where* $\beta(\pi) = \emptyset$, $\mu \neq \mathbf{ff}$, μ *is ex-free*

We state the correctness result here and include the proofs in Appendix C.

Theorem 1. *For* $\varphi \in$ LTL, $\mathcal{D}(\varphi)$ *is a limit deterministic automaton such that* $L(\mathcal{D}(\varphi)) = [\![\varphi]\!]$ *and* $\mathcal{D}(\varphi)$ *is of size at most double exponential in* φ.

The number of different formulae in $\mathcal{B}^+(\varphi)$, is at most double exponential in the size of φ, since each can be represented as a collection of subsets of subformulae of φ. $\mathit{\Pi}(\varphi)$ is simply tripartition of $\mathcal{C}(\varphi)$ which is bounded above by $3^{|\varphi|}$. And the counter can take $|\mathbb{F}(\mathcal{C}(\varphi))| + 1$ different values which is $\leq |\varphi|$. The entire state space $\mathcal{B}^+(\varphi) \times \mathcal{B}^+(\varphi) \times \mathit{\Pi}(\varphi) \times [n]$ is upper bounded by the product of these which is clearly doubly exponential.

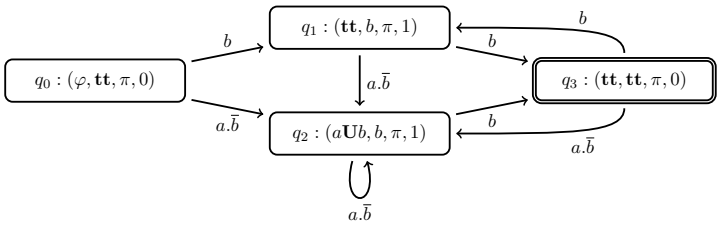


Fig. 3: Our construction for $\varphi = \mathbf{G}(a\mathbf{U}b)$.

We illustrate our construction using $\varphi = \mathbf{G}(a\mathbf{U}b)$ which is a formula outside LTL$\backslash GU$. The automaton for φ is shown in Figure 3. First note that the $\mathcal{C}(\varphi) = \{\varphi, \mathbf{F}b\}$. Next, observe that the only interesting FG-prediction is π in which $\alpha = \{\varphi\}$, $\beta = \emptyset$ and $\gamma = \{\mathbf{F}b\}$. This is because any initial state will have $\mu = \varphi$ which forces $\varphi \in \alpha$, and since predictions in α don't change, every reachable state will have $\varphi \in \alpha$ as well. As for $\mathbf{F}b$ note that the corresponding internal until $a\mathbf{U}b$ will become $\mathbf{ff}$ if $\mathbf{F}b$ is in α and thus making the derivative $\mathbf{ff}$ ($a\mathbf{U}b$ is added to the obligation at each step since $\varphi \in \alpha$ and rule **(b)**). Therefore $\mathbf{F}b$ cannot be in α, and it cannot be in β because then it would be eventually in α. So $\mathbf{F}b$ has to be in γ. Now that π is fixed, and given input σ, the obligation μ changes according to rule **(b)** as $\mu' = \nabla(\mu \wedge (a\mathbf{U}b), (\sigma, \pi))$. Similarly the carry ν changes to b if $\nu = \mathbf{tt}$ (as in q_3 to q_1/q_2) and becomes $\nu' = \nabla(\nu, (\sigma, \pi)) \vee b$

otherwise in accordance with rule **(d)**. The initial state is q_0 with $\mu = \varphi$, $\nu = \mathbf{tt}$ and counter $= 0$. The counter is incremented whenever ν becomes $\mathbf{tt}$. It is easy to see that the automaton indeed accepts $\mathbf{G}(a\,\mathbf{U}\,b)$ and is limit deterministic.

4 Efficiency

In this section we state the results regarding the efficiency of our construction for $\mathrm{LTL_D}$. We prove that there are only exponentially many reachable states in $\mathcal{D}(\varphi)$. A state $q = (\mu, \nu, \pi, n)$ of $\mathcal{D}(\varphi)$ is called reachable if there exists a valid finite run of the automaton that ends in q. A μ is said to be reachable if (μ, ν, π, n) is reachable for some choice of ν, π and n. Similarly for ν. We show that the space of reachable μ and ν is only exponentially large in the size of φ. Our approach will be to show that every reachable μ (or ν) can be expressed in a certain way, and we will count the number of different such expressions to obtain an upper bound. The expression for μ and ν relies on them being represented in DNF form and uses the syntactic definition of the derivative (Definition 15) given in the Appendix. Therefore we state only the main result and its consequence on the model checking complexity here and present the proofs in Appendix D.

Theorem 2. *For $\varphi \in \mathrm{LTL_D}$ the number of reachable states in the $\mathcal{D}(\varphi)$ is at most exponential in $|\varphi|$.*

Theorem 3. *The model checking problem for MDPs against specification in* $\mathrm{LTL_D}$ *is* EXPTIME-*complete*

Proof. The upper bound follows from our construction being of exponential size and the fact that the model checking of MDPs can be done by performing a linear time analysis of the synchronous product of the MDP and the automaton [4]. The EXPTIME hardness lower bound is from the fact that the problem is EXPTIME hard for the subfragment $\mathrm{LTL}\backslash GU$ as proved in [9].

5 Expressive power of $\mathrm{LTL_D}$

In this section we show that $\mathrm{LTL_D}$ is semantically more expressive than $\mathrm{LTL}\backslash GU$. We demonstrate that the formula $\varphi_0 = \mathbf{G}(p \vee (q\,\mathbf{U}\,r))$ which is expressible in $\mathrm{LTL_D}$, cannot be expressed by any formula in $\mathrm{LTL}\backslash GU$.

Let us fix integers $\ell, k \in \mathbb{N}$. We will use $\mathrm{LTL}_\ell(F, G)$ to denote the subfragment of LTL(F,G) where formulae have maximum height ℓ. Since $\mathbf{X}$ distributes over all other operators we assume that all the $\mathbf{X}$s are pushed inside. We use $\mathrm{LTL}_{\ell,k}\backslash GU$ to denote the fragment where formulae are built out of $\mathbf{U}$, $\wedge$, $\vee$ and $\mathrm{LTL}_\ell(F, G)$ formulae such that the number of $\mathbf{U}$s used is at most k.

Next, consider the following strings over 2^P where $P = \{p, q, r\}$:

$$u = \{p\}\{p,q\}^\ell\{p\} \quad v = \{q\}\{p,q\}^\ell\{r\} \quad w = \{q\}\{p,q\}^\ell\{p\}$$
$$s_k = (uv)^{k+1}u \qquad \sigma = (uv)^\omega \qquad \eta_k = s_k wv\sigma$$

The observation we make is that σ satisfies φ_0 but η_k does not. We state the main theorem and the corollary here and leave the details in Appendix E.

Theorem 4. $\forall\, \varphi \in \mathrm{LTL}_{\ell,k} \backslash GU \quad \sigma \vDash \varphi \implies \eta_k \vDash \varphi$

Corollary 1. *φ_0 is not expressible in $\mathrm{LTL}_{\ell,k} \backslash GU$. Also since ℓ and k are arbitrary, φ_0 is not expressible in $\mathrm{LTL} \backslash GU$.*

6 Experimental Results

We present our tool `Büchifier` (available at [1]) that implements the techniques described in this paper. `Büchifier` is the first tool to generate LDBA with provable exponential upper bounds for a large class of LTL formulae. The states (μ, ν, π, n) in our automaton described in Definition 10 (16 in Appendix), involve $\mu, \nu \in \mathcal{B}^+(\varphi)$ which are essentially sets of sets of subformulae. We view each subformula as a different proposition. We then interpret the formulae in $\mathcal{B}^+(\varphi)$ as a Boolean function on these propositions. In `Büchifier` we represent these Boolean functions symbolically using Binary Decision Diagrams (BDD). Our overall construction follows a standard approach where we begin with an initial set of states and keep adding successors to discover the entire reachable set of states. We report the number of states, number of transitions and the number of final states for the limit deterministic automata we construct.

MDP model checkers like PRISM [16], for a long time have used the translation from LTL to deterministic Rabin automata and only recently [21] have started using limit deterministic Büchi automata. As a consequence we compare the performance of our method against `Rabinizer 3` [13] (the best known tool for translating LTL to deterministic automata) and `ltl2ldba` [21] (the only other known tool for translating LTL to LDBA). `Rabinizer 3` constructs deterministic Rabin automata with generalized Rabin pairs (DGRA). The experimental results in [5, 13] report the size of DGRA using the number of states and number of acceptance pairs of the automata; the size of each Rabin pair is, unfortunately, not reported. Since the size of Rabin pairs influences the efficiency of MDP model checking, we report it here to make a meaningful comparison. We take the size of a Rabin pair to be simply the number of transitions in it. The tool `ltl2ldba` generates transition-based generalized Büchi automata (TGBA). The experimental results in [21] report the size of the TGBA using number of states and number of acceptance sets, and once again the size of each of these sets is not reported. Since their sizes also effect the model checking procedure we report them here. We take the size of an acceptance set to be simply the number of transitions in it. In Table 1 we report a head to head to comparison of `Büchifier`, `Rabinizer 3` and `ltl2ldba` on various LTL formulae.

1. The first 5 formulae are those considered in [5]; they are from the GR(1) fragment [19] of LTL. These formulae capture Boolean combination of fairness conditions for which generalized Rabin acceptance is particularly well suited. `Rabinizer 3` does well on these examples, but `Büchifier` is not far behind its competitors. The formulae are instantiations of the following templates: $g_0(j) = \wedge_{i=1}^{j}(\mathbf{GF}a_i \Rightarrow \mathbf{GF}b_i)$, $g_1(j) = \wedge_{i=1}^{j}(\mathbf{GF}a_i \Rightarrow \mathbf{GF}a_{i+1})$.

	Büchifier			Rabinizer 3			ltl2ldba		
	St	Tr	AC	St	Tr	AC	St	Tr	AC
$g_0(1)$	4	7	2	1	1	3	3	6	2 (1)
$g_0(2)$	12	23	5	1	1	8	5	14	12 (2)
$g_0(3)$	32	63	8	1	1	20	9	36	54 (3)
$g_1(2)$	12	21	5	1	1	8	5	13	11 (2)
$g_1(3)$	31	54	13	1	1	18	9	30	44 (3)
φ_1	5	7	3	5	13	40	7	23	12 (4)
φ_2	26	83	8	12	48	233	36	101	75 (2)
φ_3	13	25	3	16	128	64	21	140	129 (2)
φ_4	17	47	7	2	4	35	9	29	31 (2)
φ_5	36	111	11	12	48	330	41	133	94 (2)
$f_0(1)$	4	7	2	2	4	2	2	4	2 (1)
$f_0(2)$	14	29	5	16	74	26	4	16	16 (2)
$f_0(3)$	44	105	13	–	–	–	8	64	96 (3)
$f_0(4)$	130	369	33	–	–	–	16	256	512 (4)
$f_1(1)$	14	29	5	6	24	10	8	32	12 (1)
$f_1(2)$	130	369	33	–	–	–	64	1024	768 (2)
$f_1(3)$	1050	4801	193	–	–	–	512	32768	36K (3)
$f_2(1)$	1	1	1	2	3	2	1	1	2 (2)
$f_2(2)$	5	7	3	5	13	45	6	21	9 (3)
$f_2(3)$	19	37	7	19	109	847	19	218	28 (4)
$f_2(4)$	65	175	15	167	2529	–	93	6301	75 (5)
$f_3(1)$	2	4	1	3	7	4	1	2	3 (2)
$f_3(2)$	10	20	4	17	91	53	14	62	28 (1)
$f_3(3)$	36	78	12	–	–	–	212	2359	953 (1)
$f_3(4)$	114	288	32	–	–	–	17352	598330	167K(1)
$h(2,1)$	26	54	9	15	49	49	14	44	1(1)
$h(2,2)$	60	138	21	65	469	469	64	434	1(1)
$h(2,3)$	182	468	57	315	5119	5119	314	4892	1(1)
$h(4,1)$	80	146	36	76	250	250	75	229	1(1)
$h(4,2)$	230	464	96	990	8068	8068	989	7465	1(1)
$h(4,3)$	908	1994	348	–	–	–	–	–	–
ψ_1	35	62	9	3	6	12	3	6	8 (3)
ψ_2	7	15	3	8	39	53	2	5	18 (3)
ψ_3	29	62	8	29	116	74	62	293	27(2)
ψ_4	26	92	6	4	11	7	3	8	3(1)
ψ_5	9	58	1	5	17	9	3	9	3(1)

Table 1: A Comparison between the sizes of automata produced by `Büchifier`, `Rabinizer 3` and `ltl2ldba` on various formulae. Column St denotes the number of states, column Tr denotes the number of transitions and column AC denotes the size of the acceptance condition. Entries marked as "–" indicate that the tool failed to construct the automaton and/or the acceptance condition due to the memory limit (1GB) being exceeded.

2. The next 5 formulae are also from [5] to show how `Rabinizer 3` can effectively handle $\mathbf{X}$s. `Büchifier` has a comparable number of states and much smaller acceptance condition when compared to `Rabinizer 3` and `ltl2ldba` in all these cases. $\varphi_1 = \mathbf{G}(q \vee \mathbf{XG}p) \wedge \mathbf{G}(r \vee \mathbf{XG}\neg p)$, $\varphi_2 = (\mathbf{GF}(a \wedge \mathbf{X}^2 b) \vee \mathbf{FG}b) \wedge \mathbf{FG}(c \vee (\mathbf{X}a \wedge \mathbf{X}^2 b))$, $\varphi_3 = \mathbf{GF}(\mathbf{X}^3 a \wedge \mathbf{X}^4 b) \wedge \mathbf{GF}(b \vee \mathbf{X}c) \wedge \mathbf{GF}(c \wedge \mathbf{X}^2 a)$, $\varphi_4 = (\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}(d \vee \mathbf{X}e))$, $\varphi_5 = (\mathbf{GF}(a \wedge \mathbf{X}^2 c) \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}(d \vee (\mathbf{X}a \wedge \mathbf{X}^2 b)))$.
3. The next 15 formulae (4 groups) express a variety of natural properties, such as $\mathbf{G}(req \Rightarrow \mathbf{F}ack)$ which says that every request that is received is eventually acknowledged. As shown in the table in many of the cases `Rabinizer 3` runs out of memory (1GB) and fails to produce an automaton, and `ltl2ldba` fails to scale in comparison with `Büchifier`. The formulae in the table are instantiations of the following templates: $f_0(j) = \mathbf{G}(\wedge_{i=1}^{j}(a_i \Rightarrow \mathbf{F}b_i))$, $f_1(j) = \mathbf{G}(\wedge_{i=1}^{j}(a_i \Rightarrow (\mathbf{F}b_i \wedge \mathbf{F}c_i)))$, $f_2(j) = \mathbf{G}(\vee_{i=1}^{j}(a_i \wedge \mathbf{G}b_i))$, $f_3(j) = \mathbf{G}(\vee_{i=1}^{j}(a_i \wedge \mathbf{F}b_i))$.
4. The next 6 formulae expressible in LTL$\backslash GU$, contain multiple $\mathbf{X}$s and external $\mathbf{U}$s. `Büchifier` constructs smaller automata and is able to scale better than `ltl2ldba` in these cases as well. The formulae are instantiations of: $h(m,n) = (\mathbf{X}^m p)\,\mathbf{U}\,(q \vee (\wedge_{i=1}^{n}(a_i\,\mathbf{U}\,\mathbf{X}^m b_i)))$.
5. The last few examples are from outside of LTL$\backslash GU$. The first three are in $\mathrm{LTL_D}$ while the rest are outside $\mathrm{LTL_D}$. We found that `Büchifier` did better only in a few cases (like ψ_3), this is due to the multiplicative effect that the internal untils have on the size of the automaton. So there is scope for improvement and we believe there are several optimizations that can be done to reduce the size in such cases and leave it for future work. $\psi_1 = \mathbf{FG}((a \wedge \mathbf{X}^2 b \wedge \mathbf{GF}b)\mathbf{U}(\mathbf{G}(\mathbf{X}^2\neg c \vee \mathbf{X}^2(a \wedge b))))$, $\psi_2 = \mathbf{G}(\mathbf{F}\neg a \wedge \mathbf{F}(b \wedge \mathbf{X}c) \wedge \mathbf{GF}(a\mathbf{U}d))$, $\psi_3 = \mathbf{G}((\mathbf{X}^3 a)\mathbf{U}(b \vee \mathbf{G}c))$, $\psi_4 = \mathbf{G}((a\mathbf{U}b) \vee (c\mathbf{U}d))$, $\psi_5 = \mathbf{G}(a\mathbf{U}(b\mathbf{U}(c\mathbf{U}d)))$.

7 Conclusion

In this paper we presented a translation of formulas in LTL to limit deterministic automata, generalizing the construction from [9]. While the automata resulting from the translation can, in general, be doubly exponential in the size of the original formula, we observe that for formulas in the subfragment $\mathrm{LTL_D}$, the automaton is guaranteed to be only exponential in size. The logic $\mathrm{LTL_D}$ is a more expressive fragment than LTL$\backslash GU$, and thus our results enlarge the fragment of LTL for which small limit deterministic automata can be constructed. One consequence of our results here is a new EXPTIME algorithm for model checking MDPs against $\mathrm{LTL_D}$ formulas, improving the previously known upper bound of 2EXPTIME.

Our results in this paper, however, have not fully settled the question of when exponential sized limit deterministic automata can be constructed. We do not believe $\mathrm{LTL_D}$ to be the largest class. For example, our construction yields small automata for $\varphi = \mathbf{G}(\vee_i(p_i\,\mathbf{U}\,q_i))$, where p_i, q_i are propositions. φ is not expressible in $\mathrm{LTL_D}$. Of course we cannot have an exponential sized construction for full LTL as demonstrated by the double exponential lower bound in [21].

References

1. Büchifier. `kini2.web.engr.illinois.edu/buchifier/`.
2. Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Logic*, 5(1):1–25, January 2004.
3. T. Babiak, F. Blahoudek, M. Kretínský, and J. Strejeck. Effective translation of LTL to deterministic Rabin automata: Beyond the (F,G)-fragment. In *ATVA*, pages 24–39, 2013.
4. Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.
5. Javier Esparza and Jan Kretínský. From LTL to deterministic automata: A Safraless compositional approach. In *CAV*, pages 192–208, 2014.
6. Marc Geilen. On the construction of monitors for temporal logic properties. *Electr. Notes Theor. Comput. Sci.*, 55(2):181–199, 2001.
7. Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. Lazy probabilistic model checking without determinisation. In *CONCUR 2015,*, pages 354–367.
8. Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Proceedings of the 20th International Conference on Computer Science Logic*, CSL'06, pages 395–410.
9. D. Kini and M. Viswanathan. Limit deterministic and probabilistic automata for LTL$\setminus GU$. In *Proceedings of TACAS*, pages 628–642, 2015.
10. Dileep Kini and Mahesh Viswanathan. Probabilistic Büchi automata for LTL$\setminus GU$. Technical Report http://hdl.handle.net/2142/72686, University of Illinois at Urbana-Champaign, 2015.
11. J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theoretical Computer Science*, 363(2):182–195, 2006.
12. Joachim Klein, David Müller, Christel Baier, and Sascha Klüppelholz. *Are Good-for-Games Automata Good for Probabilistic Model Checking?*, pages 453–465. Cham, 2014.
13. Zuzana Komárková and Jan Kretínský. Rabinizer 3: Safraless translation of LTL to small deterministic automata. In *ATVA*, pages 235–241, 2014.
14. J. Kretínský and J. Esparza. Deterministic automata for the (F,G)-fragment of LTL. In *CAV*, pages 7–22, 2012.
15. J. Kretínský and R. Ledesma-Garza. Rabinizer 2: Small deterministic automata for LTL$\setminus$ GU. In *ATVA*, pages 446–450, 2013.
16. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011.
17. A. Morgenstern and K. Schneider. From LTL to symbolically represented deterministic automata. In *VMCAI*, pages 279–293, 2008.
18. N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. In *VMCAI*, pages 279–293, 2006.
19. Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. In *Verification, Model Checking, and Abstract Interpretation, 7th International Conference, VMCAI 2006*, pages 364–380, 2006.
20. M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.
21. Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínskỳ. Limit-deterministic Büchi automata for linear temporal logic. CAV, 2016.
22. Salomon Sickert and Jan Kretínský. MoChiBA: Probabilistic LTL model checking using limit-deterministic büchi automata. In *Proceedings of ATVA*, pages 130–137, 2016.

23. M. Vardi, P. Wolper, and A. P. Sistla. Reasoning about infinite computation paths. In *FOCS*, 1983.
24. M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of FOCS*, pages 327–338, 1985.

A Example

Note that every state has the formula φ present in α becuase any inital state has to evaluate φ to true and since it is a $\mathbf{G}$ formula it has to be in α and once assigned to be α it cannot be changed. Hence all states are initial. Next observe that we have two components owing to the two different γ: $\emptyset$ (in q_0, q_1) or $\{\mathbf{F}b\}$ (in q_2, q_3). In the states q_0, q_1, the subformula $\mathbf{F}b$ is in β, α respectively. We do not need a counter for this component as γ is empty and hence shown to be always 0. There is a transtion from q_0 to q_0 where the FG-prediction hasn't changed, but we need to verify that $\psi = (a \vee \mathbf{F}b)$ is true (for $\varphi \in \alpha$) at the initial q_0 which is done by observing that $\mathbf{F}b$ is predicted to be β implying the truth of ψ. The state q_0 is non-final as β is non-empty. There is a transition from q_0 to q_1 which changes the prediction for $\mathbf{F}b$ and this forces only those transitions to be enabled where b is true (if b were replaced by a more complicated formula its truth would be enforced using a combination of the input being seen and the predictions made for the smaller temporal subformulae). q_1 has a transition to q_1 only enabled when a is true because at this point $\mathbf{F}b$ is predicted to be in α and hence assumed to be false. In q_2 and q_3 the predictions don't change only the counter does. In both states since $\mathbf{F}b \in \gamma$, we get $a \vee \mathbf{F}b$ to be evaluated to be true irrespective of the input being seen therefore $\varphi \in \alpha$ is automatically checked. The only remaining thing is $\mathbf{F}b \in \gamma$ which is done using a counter. When the counter is 0 it is forced to be incremented and when the counter is non-zero (in this case 1) it is incremented when b is evaluated to be true, once again if b were replaced by a more complicated formula its truth would have been derived using the next input and the prediction at that state. It is easy to see that this automaton is indeed limit deterministic and correctly accepts $L(\varphi)$.

B Derivative and Construction

In this section we will look at the operational definition of the derivative and how it is used to define the automata construction.

We describe some terminology related to normal form representation for temporal formulae. A *term* t is a conjunction of formulae $\varphi_0, \ldots, \varphi_k$ denoted as a set $t = \{\varphi_0, \ldots, \varphi_k\}$. A term over φ is a term in which all formulae are subformulae of φ or their dependents (formulae $\mathbf{F}\psi$ for internal $\phi \,\mathbf{U}\, \psi$ and and $\mathbf{X}(\phi \,\mathbf{U}\, \psi)$ for all $\phi \,\mathbf{U}\, \psi$). The set of all terms over φ is denoted by $\mathcal{T}(\varphi)$. A *form* is a disjunction of a finitely many terms $t_1, \ldots, t_n$ represented as $\langle t_1, \ldots, t_n \rangle$. A form is said to be over φ if every term in it is over φ. Set of all such forms is denoted by $\mathcal{F}(\varphi)$. A single term t can also be interpreted as the form $\langle t \rangle$ depending on the context it is used. We use $\langle \psi \rangle$ to denote the form with a single term $\{\psi\}$. False is denoted by the empty form $\langle\rangle$ and true is represented as $\langle \emptyset \rangle$. If there are two terms t_i, t_j in a form such that $t_i \subseteq t_j$ then we can drop t_j because t_j implies t_i. For a set of terms T, let $\min(T)$ be the form consisting of the minimal (according to the subset relation) terms in T. Let $\nu_1 \cup \nu_2$ be the set of terms contained in either ν_1 or ν_2. The join of two forms ν_1, ν_2 denoted by $\nu_1 \sqcup \nu_2$ is

the form $\min(\nu_1 \cup \nu_2)$. The meet of two forms ν_1, ν_2 denoted by $\nu_1 \sqcap \nu_2$ is the form $\min(\{t_1 \cup t_2 \mid t_1 \in \nu_1, t_2 \in \nu_2\})$.

We say a term t is *ex-free* if $t \cap \Lambda_\varphi$ is empty, and form f is ex-free if each term in it is ex-free. Next, we introduce the concept of *consistency*.

Definition 11 (Consistency). *A term e is said to be locally consistent if:*

- $\phi \vee \psi \in e$ *then* $\phi \in e$ *or* $\psi \in e$.
- $\phi \wedge \psi \in e$ *then* $\phi \in e$ *and* $\psi \in e$
- $(\phi \,\mathbf{U}\, \psi \in e$ *and* $\phi \,\mathbf{U}\, \psi \notin \Lambda_\varphi)$ *then* $\mathbf{F}\psi \in e$
- $\phi \,\mathbf{U}\, \psi \in e$ *then either* $(\phi \in e$ *and* $\mathbf{X}(\phi \,\mathbf{U}\, \psi) \in e)$ *or* $\psi \in e$
- $\mathbf{ff} \notin e$

A term e is said to be consistent with input symbol $\sigma \in 2^P$ if:

- *if* $p \in e$ *then* $p \in \sigma$
- *if* $\neg p \in e$ *then* $p \notin \sigma$

A term $e \in \mathcal{T}(\varphi)$ is said to be consistent with an FG-prediction $\pi \in \Pi(\varphi)$ if:

- $\mathbf{F}\psi \in e$ *then* $\mathbf{F}\psi \notin \alpha(\pi)$
- $\mathbf{G}\psi \in e$ *then* $\mathbf{G}\psi \in \alpha(\pi)$

A term $e \in \mathcal{T}(\varphi)$ is said to be consistent with ex-choice $\lambda \subseteq \Lambda_\varphi$ if:

- $\forall\, \phi \,\mathbf{U}\, \psi \in e \cap \Lambda_\varphi$*:* $\psi \in e$ *iff* $\phi \,\mathbf{U}\, \psi \in \lambda$
- $\forall\, \phi \vee \psi \in e \cap \Lambda_\varphi$*:* $\phi \in e$ *iff* $\phi \vee \psi \in \lambda$

The notion of local consistency is an extension of the concept of "local informativeness" introduced in [6]. A term is locally consistent if every compound formula of the form $(\wedge/\vee/\mathbf{U})$ present in the term is appropriately supported by the presence of its immediate subformulae/dependents. A term that is locally consistent gives a *proof* for the satisfaction of each compound formula present in it. The proof is local in the sense that it gives a way to satisfy the current and not any future obligations that need to be met. Consider $\phi \,\mathbf{U}\, \psi$, for which to be satisfied one needs:

- ψ to hold at some point (which is expressed by the presence of dependent $\mathbf{F}\psi$), and
- either ψ holds now (presence of ψ) or ϕ holds now and $\phi \,\mathbf{U}\, \psi$ holds at the next step (presence of ϕ, $\mathbf{X}(\phi \,\mathbf{U}\, \psi)$)

For $\wedge/\vee$ we need both/one of the arguments present for the term to be locally consistent. Note that local consistency does not handle literals because the current input σ is supposed to tell us their truth; and those requirements are encoded in σ consistency constraints. Similarly π tells us the truth of $\mathbf{F}$,$\mathbf{G}$-formulae which are encoded in π consistency constraints. An ex-choice $\lambda \in \Lambda_\varphi$ also dictates additional constraints. The ex-choice λ dictates how each external $\mathbf{U}/\vee$ subformula is satisfied if it needs to be. If $\phi \,\mathbf{U}\, \psi \in \lambda$ then it must be immediately satisfied by the presence of ψ. A $\phi \vee \psi \in \lambda$ must be satisfied by the presence of ϕ. λ provides us with a resolution of choices created by external $\mathbf{U}/\vee$.

For notational simplicity, we are going to combine the three forms of constraints (input symbol, FG-prediction, ex-choice) into an extended symbol:

Definition 12 (Extended symbol). *An extended symbol for an LTL formula φ over propositions P is a triple $(\sigma, \pi, \lambda) \in 2^P \times \Pi(\varphi) \times \Lambda_\varphi$. We will use $\mathcal{E}_\varphi$ denote the space $2^P \times \Pi(\varphi) \times \Lambda_\varphi$. We say a term t is consistent with $\varepsilon \in \mathcal{E}_\varphi$ if t is consistent with each component of ε.*

(We will also sometimes refer to the pair (σ, π) as an extended symbol, this is useful when the ex-choice becomes irrelevant)

For a sequence of symbols ρ (finite or infinite) over 2^P, an *extension* is an equally long sequence $w = \{(\rho_i, \pi_i, \lambda_i)\}$ over $\mathcal{E}_\varphi$.

Next, we define the *expansion* of a term w.r.t an extended input. The expansion is a form consisting of terms that describe different ways to satisfy the given term.

Definition 13 (Expansion). *For $t \in \mathcal{T}(\varphi)$ and $\varepsilon \in \mathcal{E}_\varphi$, the expansion $\mathcal{X}(t, \varepsilon)$ is the form $\min(T)$ where T is the set of all terms e such that $t \subseteq e$, e is locally consistent, and consistent with ε. Given form ν we define the expansion $\mathcal{X}(\nu, \varepsilon)$ as $\bigsqcup_{t \in \nu} \mathcal{X}(t, \varepsilon)$*

The successor of a term t is the term consisting of all the temporal obligations that are pending in t:

Definition 14 (Successor). *The successor of a term t denoted by $\mathcal{S}(t)$ is defined as the term $\{\psi \mid \mathbf{X}(\psi) \in t\}$. For a form f, $\mathcal{S}(f)$ is defined as the form $\min(\{\mathcal{S}(t) \mid t \in f\})$.*

Next, we define the *derivative*. Here we directly define what corresponds to the weakest derivative as in Definition 9 and simply refer to it as the derivative. Given a form f, a finite sequence of input symbols ρ and an extension w of ρ: the derivate denoted by $\nabla(f, w)$ corresponds to the obligation such that if any infinite continuation ρ' satisfies it then it guarantess f to be true at $\rho\rho'$ given that w is the prefix of a *sound* extension for $\rho\rho'$. Informally, an extension is sound if the guesses made by the FG-predictions and ex-choices along the word are correct. The derivative can be seen as a generalization and a declarative version of the various unfolding operations [14, 5].

Definition 15 (Derivative). *For a form $f \in \mathcal{F}$, and extended symbol $\varepsilon \in \mathcal{E}_\varphi$ define the derivate $\nabla(f, \varepsilon)$, as the form $\mathcal{S}(\mathcal{X}(f, \varepsilon))$. We extend the definition to finite words over $\mathcal{E}_\varphi$ as: $\nabla(f, \epsilon) = f$ and $\nabla(f, \varepsilon w) = \nabla(\nabla(f, \varepsilon), w)$. For a given extension w we shall use $\nabla_i^j(f)$ as a shorthand for $\nabla(f, w[i, j])$*

Observe that the derivative of a form only consits of $\mathbf{U}$ subformulae and arguments of $\mathbf{X}$ subformulae of the given form due to the application of $\mathcal{S}$.

Now we are ready to describe the automata construction using this definition. A single state in our construction is a 4-tuple (μ, ν, π, m) where μ and ν are forms over φ, π is a FG-prediction and m is a counter. Note that forms are just a different way of representing formulae and will be more convenient when proving correctness and efficiency of the construction. The only operations a form needs to emulate are the $\wedge/\vee$ which are done by $\sqcap/\sqcup$. The construction is exactly as in Definition 10 with forms replacing formulae and the analogous boolean operations for forms.

Definition 16 (Construction). *Given a formula $\varphi \in$ LTL over propositions P, let $\mathcal{D}(\varphi)$ be the NBA (Q, δ, I, F) over the alphabet 2^P defined as follows:*

- Q *is the set* $\mathcal{F}(\varphi) \times \mathcal{F}(\varphi) \times \Pi(\varphi) \times [n]$ *where* $n = |\mathbb{F}(\mathcal{C}(\varphi))| + 1$
- δ *is the set of all transitions of the form* $(\mu, \nu, \pi, m) \xrightarrow{\sigma} (\mu', \nu', \pi', m')$ *such that*
  - **(a)** $\alpha(\pi) \subseteq \alpha(\pi')$ *and* $\gamma(\pi) = \gamma(\pi')$
  - **(b)** $\mu' = \nabla(\mu \sqcap \langle t \rangle, \varepsilon)$ *for some* $\lambda \subseteq \Lambda_\varphi$
    where $t = \{\psi \mid \mathbf{F}\psi \in \beta(\pi) \cap \alpha(\pi') \text{ or } \mathbf{G}\psi \in \alpha(\pi)\}$ *and* $\varepsilon = (\sigma, \pi, \lambda)$
  - **(c)** $m' = \begin{cases} (m+1) \,(\mathrm{mod}\, |\mathbb{F}(\gamma)| + 1) & \nu = \langle\emptyset\rangle \\ m & \text{otherwise} \end{cases}$
  - **(d)** $\nu' = \begin{cases} \langle \psi_{m'} \rangle & \nu = \langle\emptyset\rangle \\ \nabla(\nu, (\sigma, \pi)) \sqcup \langle \psi_m \rangle & \text{otherwise} \end{cases}$
    where $\{\mathbf{F}\psi_0, .., \mathbf{F}\psi_k\}$ *is an enumeration of* $\mathbb{F}(\gamma)$, $\psi_0 = \mathbf{tt}$
- I *is all states of the form* $(\langle\varphi\rangle, \langle\emptyset\rangle, \pi, 0)$
- F *is all states of the form* $(\mu, \langle\emptyset\rangle, \pi, 0)$ *where* $\beta(\pi) = \emptyset$, $\mu \neq \langle\rangle$, μ *is ex-free*

C Proof of Correctness

In order to prove correctness (Theorem 1) we define an annotation for an accepting run. The annotation is a sequence of pairs (u, v) where u and v represent expansions of terms in μ and ν respectively that are true:

Definition 17 (Annotation). *Given an accepting run of the automaton $\mathcal{D}(\varphi)$ over word ρ*

$$(\mu_0, \nu_0, \pi_0, m_0) \xrightarrow{\rho_0} \cdots (\mu_i, \nu_i, \pi_i, m_i) \xrightarrow{\rho_i} \cdots \qquad (\star)$$

an annotation is a (finite/infinite) sequence of pairs $(u_i, v_i) \in \mathcal{T} \times (\mathcal{T} \cup \{\mathsf{null}\})$ where:

$$\forall i \geq 0: \quad \lambda_i \subseteq \Lambda_\varphi \text{ such that } \mu_{i+1} = \nabla(\mu_i \sqcap \Psi_i, (\rho_i, \pi_i, \lambda_i)) \qquad (2)$$

$$\forall i \geq 0: \quad u_i \in \mathcal{X}\big(t \sqcap \Psi_i,\, \rho_i, \pi_i, \lambda_i)\big) \text{ where } t = (\mathcal{S}(u_{i-1}) \;\texttt{if}\; i > 0 \;\texttt{else}\{\varphi\})$$

$$\text{where} \quad \Psi_i = \{\psi \mid \mathbf{F}\psi \in \beta(\pi_i) \cap \alpha(\pi_{i+1}) \text{ or } \mathbf{G}\psi \in \alpha(\pi_i)\} \qquad (3)$$

$$\forall i \geq 0: \quad \emptyset \in \nu_i \iff v_i = \emptyset \qquad (4)$$

$$\forall i > 0: \quad v_{i-1} = \emptyset/\mathsf{null} \implies v_i = \mathsf{null} \text{ or } v_i \in \mathcal{X}\big(\langle \psi_{m_i} \rangle, (\rho_i, \pi_i)\big) \qquad (5)$$

$$\forall i > 0: \quad v_{i-1} \neq \emptyset/\mathsf{null} \implies v_i \in \mathcal{X}(\mathcal{S}(v_{i-1}), (\rho_i, \pi_i)) \qquad (6)$$

Lemma 1. *For a given accepting run of $\mathcal{D}(\varphi)$ $(\star)$*

$$\forall n \geq 0;\ \forall s \in \mu_n;\ \forall x \in \mathcal{X}(s \sqcap \Psi_n, (\rho_n, \pi_n, \lambda_n));\ \forall y \in \mathcal{X}(\nu_n, (\rho_n, \pi_n))$$

there exists an annotation of length $n{+}1$ ending in (x, y)

Proof. First note that the requirements for sequence of u_i and those for v_i in an annotation are independent. We show the existence of a sequence $\{u_i\}_{0\leq i\leq n}$ ending in x and sequence $\{v_i\}_{0\leq i\leq n}$ ending in y each satisfying their respective conditions. Combining the sequences will then gives us the required annotation. First note that λ_i in condition 2 is given by the transtions taken in the accepting run. We perform indunction on n to construct the sequences.

Base Case: when $n = 0$ we have $\mu_n = \langle\varphi\rangle$ (initial state condition), so s can only be $\{\varphi\}$ and hence $x \in \mathcal{X}(\{\varphi\} \sqcap \Psi_0, (\rho_0, \pi_0, \lambda_n))$ which satisfies condition 3. For $n = 0$ we have $\nu_n = \langle\emptyset\rangle$, so $y = \emptyset$ which would satisfy condition 4. Conditions 5 and 6 do not apply for the base case (see quantification on the conditions).

Inductive Case: We assume the statement is true for n upto i and prove it for $n = i + 1$.

Extending u: Let t be a term in μ_i such that the quantified s is contained in $\nabla(t \sqcap \Psi_i, (\rho_i, \pi_i, \lambda_i)) = \mathcal{S}(\mathcal{X}(t \sqcap \Psi_i, (\rho_i, \pi_i, \lambda_i)))$. Such a term t should exist by the fact that $s \in \mu_{i+1}$ and how μ_{i+1} is derived from μ_i in Rule **(b)**. Now let x' be the term in $\mathcal{X}(t \sqcap \Psi_i, (\rho_i, \pi_i, \lambda_i))$ for which $\mathcal{S}(x') = s$. By the inductive hypothesis we have $\{u_j\}_{0\leq j\leq i}$ that ends in x' which satisfies the condition 3. We append x to this sequence to obtain the required sequence of length $i + 1$

Extending v: Consider the case when $\nu_i = \langle\emptyset\rangle$, here we know $\nu_{i+1} = \langle\psi_{m_{i+1}}\rangle$ implying $y \in \mathcal{X}(\langle\psi_{m_{i+1}}\rangle, (\rho_{i+1}, \pi_{i+1}))$. Condition 4 is satisfied for $v_{n+1} = y$ because y is non-empty, as $\psi_{n_{i+1}}$ is contained in y. For $\nu_i = \langle\emptyset\rangle$ we need to check condition 5, which is satisfied because $y \in \mathcal{X}(\langle\psi_{m_{i+1}}\rangle, (\rho_{i+1}, \pi_{i+1}))$. Next, consider the case when $\nu_i \neq \langle\emptyset\rangle$, here we know $\nu_{i+1} = \nabla(\nu_i, (\rho_i, \pi_i)) \sqcup \langle\psi_{m_i}\rangle$. Now suppose r be the term in ν_{i+1} for which $y \in \mathcal{X}(r, (\rho_{i+1}, \pi_{i+1}))$. If $r \in \nabla(\nu_i, (\rho_i, \pi_i))(= \mathcal{S}(\mathcal{X}(\nu_i, (\rho_i, \pi_i))))$ let $y' \in \mathcal{X}(\nu_i, (\rho_i, \pi_i))$ for which $r = \mathcal{S}(y')$. By the induction hypothesis we have $\{v_j\}_{0\leq j\leq i}$ that ends in y'. We append y to this sequence to obtain the required sequence of length $i{+}1$. If $r = \{\psi_{m_i}\}$, consider the last index k where $n_k \neq n_i$. (If such a k does not exist, then all m_i have to be zero in which case we have $v_i = \emptyset$ for all i). Here ν_k has to be $\langle\emptyset\rangle$ owing to the fact that counter changes only when ν becomes $\langle\emptyset\rangle$. By induction hypothesis there exists a sequence $\{v_j\}_{0\leq j\leq k}$ ending in $\emptyset$. Appending $\mathsf{null}^{(k-i-1)}y$ to this gives us the required sequence.

Corollary 2. *Every accepting run of $\mathcal{D}(\varphi)$ has an infinite annotation*

Proof. Note that in an accepting run $\mu_i \neq \emptyset$ for any i, which means there will always exist $s \in \mu_i$ such that $\mathcal{X}(s \cup \Psi_i, (\rho_i, \pi_i, \lambda_i)) \neq \emptyset$ (otherwise $\mu_{i+1} = \emptyset$). In an accepting run we also know that $\nu_i = \langle\emptyset\rangle$ for infinitely many i. Hence using Lemma (1) we have finite sized annotations of arbitrarily large lengths. Note that every prefix of an annotation is also a valid annotation. We can arrange all these annotations and their prefixes in an infinite rooted tree where the root is the empty annotation, and there is an edge between two annotations if one of them is a prefix of the other obtained by removing the last element. Every node in this tree has finite degree because the space of each element in this sequence

is finite ($\mathcal{T} \times (\mathcal{T} \cup \{\mathsf{null}\})$). Finally we use König's lemma to obtain an infinite path in this tree which gives us the required infinite annotation.

Let $Set : \mathcal{T} \cup \{\mathsf{null}\} \to \mathcal{T}$ be the function which maps every $\mathcal{T}$ to itself and maps null to the empty term $\emptyset$.

Lemma 2. *For an accepting run ($\star$) with an infinite annotation $\{(u_i, v_i)\}_{i\geq 0}$, it is the case that $\forall i \geq 0\ \forall\, \theta \in (u_i \cup Set(v_i)) : \rho[i] \vDash \theta$*

Proof. We perform induction on the size of θ. We shall use the fact that $S_i = u_i \cup Set(v_i)$ is a term that is locally consistent and consistent with π_i and w_i.

For the base case when θ is a literal $p/\neg p$ it has to be the case that $p/\neg p$ is respectively true for $\rho[i]$ because it is contained in a term that is consistent with w_i.

If θ is of the form $\phi \vee / \wedge \psi$ then using local consistency we get that either (or both) of ϕ or ψ is present in S_i, and hence true at $\rho[i]$ due to the inductive hypothesis which gives us the truth of θ at $\rho[i]$.

If θ is $\phi\ \mathbf{U}\ \psi$: if $\theta \notin \Lambda_\varphi$ then local consistency tells us that $\mathbf{F}\psi \in S_i$ and using the induction hypothesis we get that $\mathbf{F}\psi$ is true at $\rho[i]$. Let $j \geq i$ the smallest index such that $\rho[j] \vDash \psi$. $\forall k : i \leq k < j$ we can inductively prove (this is a separate induction on k) that $\phi, \mathbf{X}(\phi\ \mathbf{U}\ \psi) \in S_k$ using the facts: that ψ is absent from all such S_k (induction hypothesis), local consistency of S_k and that $\mathbf{X}(\phi\ \mathbf{U}\ \psi)$ transfers $\phi\ \mathbf{U}\ \psi$ to the next step (definition of annotation and $\mathcal{S}$). If $\theta \in \Lambda_\varphi$ then there is a $j \geq i$ such that θ is absent in μ_j (because once a final state is reached μ becomes ex-free), picking the smallest such j one obtains the point at which ψ is true which can be proved by induction on j like we did in previous case.

If θ is $\mathbf{F}\psi$ then using the fact that S_i is π_i consistent we can infer that $\mathbf{F}\psi \in \beta(\pi_i) \cup \gamma(\pi_i)$. If $\psi \in \beta(\pi_i)$ then $\exists j \geq i$ such that $\mathbf{F}\psi \in \beta(\pi_j) \cap \alpha(\pi_{j+1})$ (due to **(a)**). Now we know $\psi \in \Psi_j$ and hence $\psi \in S_j$ (from definition of annotation). Using the induction hypothesis we get $\rho[j] \vDash \psi$ and hence $\rho[i] \vDash \mathbf{F}\psi$. If $\psi \in \gamma(\pi_i)$ then we know that the counter will eventually (say at $j \geq i$) become the index m_j corresponding to ψ in γ (γ doesn't change along a run, see **(a)**). Let j be the smallest such index. Consider the smallest $k \geq j$ for which $v_k \neq \mathsf{null}$. m_k has to be equal to m_j because the counter cannot change while v_i is empty. From property (5) of annotations we get that $v_k \in \mathcal{X}(\{\psi\}, \pi_k, w_k)$. This tells us that $\psi \in S_k$ and by the induction hypothesis we have $\rho[k] \vDash \psi$ thus proving $\rho[i] \vDash \mathbf{F}\psi$.

If θ is $\mathbf{G}\psi$ then using the fact that S_i is consistent with π_i we infer that $\mathbf{G}\psi \in \alpha(\pi_i)$. Now using **(a)** we infer that $\mathbf{G}\psi \in \alpha(\pi_j)$ for all $j \geq i$. Using property (3) of annotations we get that $\psi \in u_j (\subseteq S_j)$. Applying the induction hypothesis we get that $\rho[j] \vDash \psi$ for all $j \geq i$ and hence $\rho[i] \vDash \mathbf{G}\psi$.

Corollary 3. *If w has an accepting run in $\mathcal{D}(\varphi)$ then $w \vDash \varphi$.*

Proposition 1. *If $w \vDash \varphi$ then w has an accepting run in $\mathcal{D}(\varphi)$*

Proof. Define the run $(\mu_i, \nu_i, \pi_i, m_i)$ as follows. Let π_i be such that

$$\alpha(\pi_i) = \{\mathbf{G}\psi \in \mathcal{C}(\varphi) \mid \rho[i] \vDash \mathbf{G}\psi\} \cup \{\mathbf{F}\psi \in \mathcal{C}(\varphi) \mid \rho[i] \nvDash \mathbf{F}\psi\}$$
$$\gamma(\pi_i) = \{\mathbf{G}\psi \in \mathcal{C}(\varphi) \mid \rho[i] \nvDash \mathbf{FG}\psi\} \cup \{\mathbf{F}\psi \in \mathcal{C}(\varphi) \mid \rho[i] \vDash \mathbf{GF}\psi\}$$

Define λ_i as $\{\, \psi \,\mathbf{U}\, \phi,\ \phi \vee \psi \in \Lambda_\varphi \mid \rho[i] \vDash \phi\}$. Fixing the sequences π_i, λ_i resolves all the non-determinism present along a run, we can then define μ_i, ν_i, m_i as described by the initial state and the transition relation. We will be then left to check that the run is indeed an accepting one.

We first show that $\forall i\, \mu_i \neq \langle\rangle$. We prove a stronger statement $\exists\, t \in \mu_n,\ \forall\, \psi \in t : \rho[n] \vDash t$ by induction on n. Base case is trivial as $\mu_0 = \langle\varphi\rangle$ for the initial state. Next consider the inductive case where $n = i + 1$. Let t_i be term in μ_i which is true according to the induction hypothesis. We can then recursively construct the set $e \supseteq (t_i \cup \Psi_i)$ such that all formulae we add to e are true by looking at the truth of the immediate subformulae of the formulae present in e. For example consider $\phi \,\mathbf{U}\, \psi \in e' = (e \cap \Lambda_\varphi)$, if $\phi \,\mathbf{U}\, \psi \in \lambda_i$ then we add ψ to e as we know that $\rho[i] \vDash \psi$ by definition of λ_i, otherwise we add $\phi, \mathbf{X}(\phi \,\mathbf{U}\, \psi)$ to e to ensure local consistency. Note that the formulae we add will be true at $\rho[i]$. Whenever there is a choice as to how we can make a certain formula true then we non-deterministically pick one. We do so for every formulae in e. The set e constructed will be locally consistent, consistent with π_i (by it's definition and the fact that all formulae in e are true at $\rho[i]$), consistent with w_i (by the fact all formulae in e hold at $\rho[i]$) and λ_i consistent (due to the way we construct e). Consider all minimal such e (among all those that can be constructed this way) and they would have to be present in $\mathcal{X}(t \cup \Psi_i, (w_i, \pi_i, \lambda_i))$. Then pick t_{i+1} as $\mathcal{S}(e)$ which is a minimal among all such e (to ensure $t_{i+1} \in \mu_{i+1}$), and observe that every formulae in $\mathcal{S}(e)$ is true at $\rho[i{+}1]$ by semantics of $\mathbf{X}$ and the fact that all formulae in e hold at $\rho[i]$.

Next we show that ν_i is $\langle\emptyset\rangle$ for infinitely many i. In order to do so we define the metric f which for a given word ρ and formula ψ gives us a number $f(\rho, \psi)$ which gives us an upper bound on the number of steps it takes ν to become $\langle\emptyset\rangle$ along ρ if it were to start at $\langle\psi\rangle$.

Definition 18. *For a formula φ and a word ρ define $f(\rho, \varphi) \in \mathbb{N} \cup \{\infty\}$ such that: $f(\rho, \varphi) = \infty$ if $\rho \nvDash \varphi$ otherwise we recursively define it as follows*

$$\begin{aligned}
f(\rho, \varphi) &= 1 && \text{if } \varphi \text{ is } p/\neg p/\mathbf{F}\psi/\mathbf{G}\psi \\
f(\rho, \mathbf{X}\phi) &= 1 + f(\rho[1], \phi) \\
f(\rho, \phi \wedge \psi) &= \max(f(\rho, \phi), f(\rho, \psi)) \\
f(\rho, \phi \vee \psi) &= \min(f(\rho, \phi), f(\rho, \psi)) \\
f(\rho, \phi \,\mathbf{U}\, \psi) &= \max_{j<i}(j + f(\rho[j], \phi), i + f(\rho[i], \psi)) && i \text{ is min s.t } \rho[i] \vDash \psi
\end{aligned}$$

We also extend this definition for a term t: define $f(\rho, t)$ as $\max_{\varphi \in t} f(\rho, \varphi)$

Next, using the above metric we show the following statement which claims that if a term t is true at $\rho[j]$ then taking the derivative of t for sufficiently many

times will yield the $\emptyset$ term.

$$\forall j \, \forall t \, \forall k \; (\forall \psi \in t : \rho[j] \vDash \psi) \wedge (k = j + f(\rho[j], t)) \Rightarrow \emptyset \in \nabla(\{t\}, \pi[j,k], w[j,k]) \quad (7)$$

We prove this using induction on k. Consider a term t that holds for $\rho[i]$. One can show that $\exists t' \in \nabla(t, (\rho_i, \pi_i))$ such that t' is true for $\rho[i+1]$ and $f(\rho[i{+}1], t') < f(\rho[i], t)$. Using this along with the induction hypothesis will prove the inductive case. For the base case when $f(\rho[i], t)$ is 1 the formulae in t can only be boolean combinations of $p/\neg p/\mathbf{F}\psi/\mathbf{G}\psi$ which do not produce any obligations ($\mathbf{X}$ formulae) which implies that $\emptyset \in \nabla(t, (\rho_i, \pi_i))$.

Now, for contradiction assume ν is $\langle\emptyset\rangle$ finitely many times, and let i be the index succeeding the last point where $\nu = \langle\emptyset\rangle$. Let ψ denote the formula ψ_{m_i}. Since $\mathbf{F}\psi \in \gamma(\pi_i)$ we know that ψ holds infinitely often by definition of $\gamma(\pi_i)$, so consider a point $j \geq i$ such that $\rho[j] \vDash \psi$. Using (7) we get that that within $f(\rho[j], \psi)$ steps ν would have to be $\langle\emptyset\rangle$ which is a contradiction.

Next, we note that $\beta(\pi_i)$ is empty for sufficiently large i, this is the same observation as made in [10]. In order to prove μ eventually becomes ex-free we can once again use a metric similar to f to argue that within finite steps every external subformula disappears from μ.

Proposition 2. *$\mathcal{D}(\varphi)$ is limit deterministic*

Proof. First note that ν' and n are deterministically updated, they only depend on ν, π which is a part of the state and σ which is the current input symbol. The only non-determinism in the automaton comes from the evolution of μ and π. In a final state, the π cannot change any further due to monotonicity and $\beta(\pi)$ being empty. In a final state μ becomes ex-free and remains ex-free from then on because the formula introduced in μ come from Ψ all of which are internal. This implies that the ex-choice λ does not play a role in determining μ (as it is ex-free) and hence μ is also updated deterministically from then on.

D Proof of Efficiency

In order to prove Theorem 2, we first observe identities about the derivative that we shall use in proofs appearing later in this Section.

Lemma 3. *For forms A and B and extended symbol ε: $\nabla(A \sqcup / \sqcap B, \varepsilon) = \nabla(A, \varepsilon) \sqcup / \sqcap \nabla(B, \varepsilon)$*

Lemma 4. *Given $\varphi \in \mathrm{LTL}\backslash GU$, a subformula ψ, and an extended input symbol $\varepsilon \in \mathcal{E}_\varphi$, the derivative of ψ, $\nabla(\psi)$ (short for $\nabla(\psi, \varepsilon)$) satisfies the following identities depending on the structure of ψ:*

$$\nabla(\psi_1 \mathbf{U} \psi_2) = \begin{cases} \nabla(\psi_2) \sqcup (\nabla(\psi_1) \sqcap \langle \psi_1 \mathbf{U} \psi_2 \rangle) & \text{if } \psi_1 \mathbf{U} \psi_2 \notin \Lambda_\varphi \\ \nabla(\psi_2) & \text{if } \psi_1 \mathbf{U} \psi_2 \in \lambda \\ \nabla(\psi_1) \sqcap \langle \psi_1 \mathbf{U} \psi_2 \rangle & \text{otherwise} \end{cases}$$

$$\nabla(\psi_1 \vee \psi_2) = \begin{cases} \nabla(\psi_1) \sqcup \nabla(\psi_2) & \text{if } \psi_1 \vee \psi_2 \notin \Lambda_\varphi \\ \nabla(\psi_1) & \text{if } \psi_1 \vee \psi_2 \in \lambda \\ \nabla(\psi_2) & \text{otherwise} \end{cases}$$

$$\nabla(\psi_1 \wedge \psi_2) = \nabla(\psi_1) \sqcap \nabla(\psi_2)$$

$$\nabla(\mathbf{F}\psi) = \langle\rangle \;\; \text{if } \mathbf{F}\psi \in \alpha(\pi) \text{ else } \langle\emptyset\rangle \qquad \nabla(p) = \langle\emptyset\rangle \;\; \text{if } p \in \sigma \text{ else } \langle\rangle$$

$$\nabla(\mathbf{G}\psi) = \langle\emptyset\rangle \;\; \text{if } \mathbf{G}\psi \in \alpha(\pi) \text{ else } \langle\rangle \qquad \nabla(\neg p) = \langle\rangle \;\; \text{if } p \in \sigma \text{ else } \langle\emptyset\rangle$$

Next, we observe that the terms in the derivative of an LTL(F,G) formula φ w.r.t a word of length k consist only of subformulae at depth k in φ, and hence derivatives of φ of order greater than h are either true or false where h is the height of φ. The lemma can be proved by induction on k.

Lemma 5. *For $\varphi \in$ LTL(F,G), $w \in \mathcal{E}^k_\varphi$, $t \in \nabla(\langle\varphi\rangle, w)$: every $\phi \in t$ is such that ϕ is a formula at depth k within φ.*
Corollary: If $\varphi \in$ LTL(F,G) is of height $h \geq 0$, then $\nabla(\langle\varphi\rangle, w) \in \{\langle\rangle, \langle\emptyset\rangle\}$ for all $|w| > h$.

Now we see how to represent the space of reachable ν and μ. Let us fix a formula φ and an infinite sequence of extended symbols w. Note that the ν component of a run cycles through $\mathbb{F}(\gamma)$. When ν becomes $\langle\emptyset\rangle$ it moves to the next $\mathbf{F}\psi$ in $\mathbb{F}(\gamma)$. $\mathbb{F}(\gamma)$ is at most as large as the given formula, hence it suffices to show a bound on reachable ν for a single $\mathbf{F}\psi$. With this in mind we fix ψ and define ν_i inductively as follows: $\nu_0 = \langle\psi\rangle$ and $\nu_{i+1} = \nabla(\nu_i, w_i) \sqcup \langle\psi\rangle$. For μ define $\mu_0 = \langle\varphi\rangle$ and $\mu_{i+1} = \nabla(\mu_i \sqcap \Psi_i, w_i)$ where $\Psi_i = \{\psi \mid \mathbf{F}\psi \in \beta(\pi_i) \cap \alpha(\pi_{i+1}) \text{ or } \mathbf{G}\psi \in \alpha(\pi_i)\}$. The sequence μ_i describes the μ component of a run. Our aim is to find a representation for ν_i, μ_i and show that the number of different possible representations is exponential. The following Proposition proved using Lemma 5 gives us a representation.

Proposition 3. *If $\psi \in$ LTL(F,G) and $\varphi \in \mathrm{LTL}_\mathrm{D}$ are of height k, and $l = \max(i-k, 0)$*

$$\nu_i = \bigsqcup_{j=l}^{i} \nabla^i_j(\langle\psi\rangle) \text{ or } \langle\emptyset\rangle \qquad \mu_i = \bigsqcap_{j=l}^{i-1} \nabla^i_j(\Psi_j) \;\sqcap\; \nabla^i_0(\langle\varphi\rangle) \text{ or } \langle\rangle$$

Proof. Consider ν_i, we can prove that it is $\sqcup_{j=0}^{i} \nabla^i_j(\psi)$ by inducting on i and using Lemma 3. Then we observe that the first $l-1$ elements are either true ($\langle\emptyset\rangle$) or false ($\langle\rangle$) due to Lemma 5. The representation above follows immediately. For μ_i the structure can be derived in a similar fashion the only difference is the extra term $\nabla^i_0(\varphi)$ that arises due to the initial condition. □

Note if ν_i is not $\langle\emptyset\rangle$ then it is completely determined by the substring of extended symbols $w[l,i]$. There are at most $(2^{|P|}.3^{|\varphi|}.2^{|\Lambda_\varphi|})^k$ such substrings for each length and there are k different lengths. Hence we get that ν_i can take on at most exponentially different values. Observe that in μ_i the part $\prod_{j=l}^{i-1}\nabla_j^i(\Psi_j)$ can take exponentially many different values once again due to the fact that it only depends on the $w[l,i]$. What remains to be seen that $\nabla_0^i(\langle\varphi\rangle)$ takes on at most exponentially different values. The next Lemma states that every derivative of φ over $w[0,i]$ can be expressed as the derivative of a single term t over $w[l,i]$.

Lemma 6. *For $\varphi \in \mathrm{LTL_D}$ of height k and $l = max(i-k,0)$, it is the case that $\nabla_0^i(\langle\varphi\rangle) = \nabla_l^i(t)$ for some $t \in \mathcal{T}(\varphi)$*

Proof. It is sufficient to prove that for any formula $\varphi \in \mathrm{LTL_D}$ and any given extension w, there exists $t \in \mathcal{T}(\varphi)$ such that $\nabla_0^{k+1}(\varphi) = \nabla_1^{k+1}(t)$. We perform structural induction on φ. If the formula φ is an internal subformula of the form $\psi\ \mathbf{U}\ \phi$ then depending on if $\nabla_0^k(\phi)$ is true or false (Lemma 5) we get that $\nabla_0^{k+1}(\varphi)$ is either true or $\nabla_0^{k+1}(\psi) \sqcap \nabla_1^{k+1}(\varphi)$ (by Lemma 4) and then by induction hypothesis we have our term. Simlar argument would work for an internal $\vee$. For $\wedge$ (internal or external) use induction hypothesis to obtain the terms for the individual arguments and the required term would be their $\sqcap$. For $\mathbf{X}(\psi)$, $\nabla_0^{k+1}(\mathbf{X}\psi) = \nabla_1^{k+1}(\psi)$. For literals and $\mathbf{F},\mathbf{G}$-formulae, the term is either true or false depending on σ_0 and π_0. For external $\mathbf{U}$ we refer to λ_0 to choose between immediate and delayed satsifaction of the $\mathbf{U}$ and on the basis of that we can get the term by induction hypothesis on the appropriate arguments. A similar argument would work for $\vee$.

The space of $\nabla(t, w[l,i])$ at most exponential because there are only $2^{|\varphi|}$ many different terms t and exponentially many different sequences of the form $w[l,i]$ which gives us the required bound on $\nabla_0^i(\langle\varphi\rangle)$.

E Proofs of Inexpressivity

In order to prove Theorem 4 we will need a proposition and a lemma. We will use $\mathrm{suf}(w)$ to denote the set of all suffixes of the word w.

Proposition 4. *For any $\rho \in \mathrm{suf}(\eta_k)$ one of the following holds*

1. *Either $\rho \in \mathrm{suf}(\sigma)$, or*
2. *$\exists\, x \in \mathrm{suf}(s_k)$ such that $\rho = xwv\sigma$*

For $\rho \in \mathrm{suf}(\eta_k)\backslash\mathrm{suf}(\sigma)$, i.e., ρ is of the form $xwv\sigma$ where $x \in \mathrm{suf}(s)$, let $cut_w(xwv\sigma) = xv\sigma \in \mathrm{suf}(\sigma)$. What we are trying to say is that for all suffixes of η_k that are not suffixes of σ, cut_w removes the substring w. Next we show that every suffix of η_k is logically equivalent (w.r.t $\mathrm{LTL}_\ell(F,G)$) to some suffix of σ.

Lemma 7. *For every $\psi \in \mathrm{LTL}_\ell(F,G)$, for any k and any $\rho \in \mathrm{suf}(\eta_k)\backslash\mathrm{suf}(\sigma)$:*
$\rho \vDash \psi$ *iff* $cut_w(\rho) \vDash \psi$

Proof. ρ is of the form $xwv\sigma$. We perform induction on $|x|$ to prove the required statement.

Base Case: $x = \epsilon$, i.e $\rho = wv\sigma$. We prove the base case by induction on ψ. Observe that for every ψ of the form $\mathbf{X}^i a$ where $a \in \{p, q, r\}$ and $0 \leq i \leq \ell$, we have $wv\sigma \vDash \psi$ iff $v\sigma \vDash \psi$. For the inductive step the only intersting cases are when ψ is $\mathbf{F}$ or $\mathbf{G}$ formula. Consider $\psi = \mathbf{F}\psi_1$

$$\begin{aligned}
v\sigma \vDash \mathbf{F}\psi_1 &\implies \exists y \in \text{suf}(v\sigma) \text{ s.t } y \vDash \psi_1 \\
&\implies wv\sigma \vDash \mathbf{F}\psi_1 \text{ becaue } y \in \text{suf}(wv\sigma) \\
wv\sigma \vDash \mathbf{F}\psi_1 &\implies \text{either (a) } wv\sigma \vDash \psi_1 \Rightarrow v\sigma \vDash \psi_1 \text{ (ind hyp)} \Rightarrow v\sigma \vDash \mathbf{F}\psi_1 \\
&\quad \text{or (b) } \exists y \in \text{suf}(wv\sigma)\backslash\{wv\sigma\} \text{ s.t } y \vDash \psi_1 \\
&\quad\quad \Rightarrow v\sigma \vDash \mathbf{F}\psi_1 \text{ because } y \in \text{suf}(v\sigma)
\end{aligned}$$

We continue the induction on ψ by considering the case when $\psi = \mathbf{G}\psi_1$

$$\begin{aligned}
v\sigma \vDash \mathbf{G}\psi_1 &\implies \forall y \in \text{suf}(v\sigma), y \vDash \psi_1 \\
&\implies \forall y \in \text{suf}(wv\sigma)\backslash\{wv\sigma\}, y \vDash \psi_1 \text{ and} \\
&\quad\quad wv\sigma \vDash \psi_1 (\text{since } v\sigma \vDash \psi_1 \text{ and ind hyp}) \\
&\implies wv\sigma \vDash \mathbf{G}\psi_1 \\
wv\sigma \vDash \mathbf{G}\psi_1 &\implies \forall y \in \text{suf}(wv\sigma), y \vDash \psi_1 \\
&\implies \forall y \in \text{suf}(v\sigma), y \vDash \psi_1 (\text{suf}(v\sigma) \subseteq \text{suf}(wv\sigma)) \\
&\implies v\sigma \vDash \mathbf{G}\psi_1
\end{aligned}$$

Returning back to the induction on x consider $\rho = xwv\sigma$ where $x = ay$ with $a = \{p\}$, $\{q\}$, $\{r\}$ or $\{p, q\}$. The proof is again by structural induction on ψ and similar to the base case hence we omit it.

In order to prove Theorem 4 we prove the following stronger statement: For $\varphi \in \text{LTL}_{\ell,k}\backslash GU$, $x \in \text{suf}(uv)$, $j \geq k$: if $x\sigma \vDash \varphi$ then $x(uv)^j uwv\sigma \vDash \varphi$. We perform induction on k. The base case is when $k = 0$ i.e φ has no $\mathbf{U}$, and it directly follows from the previous Lemma. For the inductive case consider $k = n + 1$. The interesting case is when φ is of the form $\psi_1 \mathbf{U} \psi_2$. The first position i along $x\sigma$ where ψ_2 holds has to be $< |uv|$. Consider the suffix of $x(uv)^m uwv\sigma$ at position i , using induction hypothesis we can conclude that ψ_2 holds at that position. Similarly for every prefix of $x(uv)^m uwv\sigma$ that begins before position i we can conclude that it satisfies ψ_1 using induction hypothesis.