Solving MDPs with LTLf+ and PPLTL+ Temporal Objectives

Giuseppe De Giacomo[1] , **Yong Li**[2] [*] , **Sven Schewe**[3] , **Christoph Weinhuber**[1] , **Pian Yu**[4] [*]
[1] Department of Computer Science, University of Oxford, UK
[2] Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software Chinese Academy of Sciences, PRC
[3] Department of Computer Science, University of Liverpool, UK
[4] Department of Computer Science, University College London, UK
giuseppe.degiacomo@cs.ox.ac.uk, liyong@ios.ac.cn, sven.schewe@liverpool.ac.uk, christoph.weinhuber@cs.ox.ac.uk, pian.yu@ucl.ac.uk

Abstract

The temporal logics LTLf+ and PPLTL+ have recently been proposed to express objectives over infinite traces. These logics are appealing because they match the expressive power of LTL on infinite traces while enabling efficient DFA-based techniques, which have been crucial to the scalability of reactive synthesis and adversarial planning in LTLf and PPLTL over finite traces. In this paper, we demonstrate that these logics are also highly effective in the context of MDPs. Introducing a technique tailored for probabilistic systems, we leverage the benefits of efficient DFA-based methods and compositionality. This approach is simpler than its non-probabilistic counterparts in reactive synthesis and adversarial planning, as it accommodates a controlled form of nondeterminism ("good for MDPs") in the automata when transitioning from finite to infinite traces. Notably, by exploiting compositionality, our solution is both implementation-friendly and well-suited for straightforward symbolic implementations.

1 Introduction

Temporal logics are widely used as specification languages in reactive synthesis and adversarial planning [Baier and Katoen, 2008; Camacho *et al.*, 2019]. Among these, linear temporal logic (LTL) [Pnueli, 1977] is perhaps the most commonly used. LTL is a formalism used to specify and reason about the temporal behaviour of systems over infinite traces. It has been extensively employed as a specification mechanism for temporally extended goals, as well as for expressing preferences and soft constraints in various fields, including business processes, robotics, and AI [Bienvenu *et al.*, 2011; Maggi *et al.*, 2011; Fainekos *et al.*, 2009]. Linear temporal logic over finite traces (LTLf) [Baier and McIlraith, 2006; De Giacomo and Vardi, 2013; De Giacomo and Vardi, 2015] is a variant of LTL with the same syntax but it is interpreted over finite instead of infinite traces. PPLTL is the pure-past version of LTLf and scans the trace backwards from the end towards the beginning [De Giacomo *et al.*, 2020]. It is well-established that strategy synthesis for LTLf and PPLTL can be derived from deterministic finite automata (DFA), thereby avoiding the challenges associated with determinising automata for infinite traces, typical of LTL reactive synthesis.

The temporal logics LTLf+ and PPLTL+ have recently been proposed to express objectives over infinite traces [Aminof *et al.*, 2024]. These logics are directly based on Manna and Pnueli's hierarchy of temporal properties [Manna and Pnueli, 1990]. This hierarchy categorizes temporal properties on infinite traces in *four classes*, which are obtained by requiring that a finite trace property holds for *some prefixes* of infinite traces ("guarantee"), *all prefixes* ("safety"), *infinitely many prefixes* ("recurrence") and for *all but finitely many prefixes* ("persistence"). Notably every LTL property can be expressed as a Boolean combination of these four kinds of properties [Manna and Pnueli, 1990]. LTLf+ and PPLTL+ use respectively LTLf and PPLTL to express properties over finite traces and obtain the four basic kinds of infinite trace properties of the Manna and Pnueli's hierarchy. This makes them particularly interesting from the computational point of view. While they retain the expressive power of LTL on infinite traces, they enable the lifting of DFA-based techniques developed for LTLf and PPLTL to obtain deterministic automata on infinite traces corresponding to formulas, thus avoiding determinisation of Büchi automata, which is known to be a notorious computational bottleneck. As a result, LTLf+ and PPLTL+ are particularly promising for a number of tasks, such as reactive synthesis [Pnueli and Rosner, 1989; Finkbeiner, 2016], supervisory control for temporal properties [Ehlers *et al.*, 2017], and planning for temporally extended goals in nondeterministic domains [Bacchus and Kabanza, 1998; De Giacomo and Rubin, 2018]. All these tasks require to obtain from the temporal formula a *deterministic automaton on infinite traces*, to be used as a game arena over which a strategy can be computed to achieve the required property. LTLf+ and PPLTL+ excel at these tasks by enabling simple arena construction through the Cartesian product of DFAs, corresponding to the finite trace (LTLf/PPLTL) components in the LTLf+/PPLTL+ formula. On the other hand, the game to be solved over this arena is an Emerson-Lei game [Emerson and Lei, 1987], which requires quite sophisticated techniques [Hausmann *et al.*, 2024].

[*]Corresponding author

In this paper, we demonstrate that these logics are even more effective in the context of MDPs. Traditionally, deterministic Rabin automata have been the standard choice for representing LTL specifications in MDPs [Baier and Katoen, 2008]. Due to the probabilistic nature of MDPs, the automaton for temporal specifications does not need to be entirely deterministic. State-of-the-art MDP synthesis methods use a restricted form of Büchi automata called *Limit-Deterministic Büchi Automata* (LDBAs) for LTL [Hahn *et al.*, 2015; Sickert *et al.*, 2016; Shao and Kwiatkowska, 2023]. Recent work has shown that an even more relaxed form of nondeterminism, termed "*good for MDPs*" (GFM), can be effectively used for MDP synthesis [Hahn *et al.*, 2020; Schewe *et al.*, 2023].

Since this relaxed nondeterminism enables more succinct representations of temporal specifications, we adopt GFM automata in this work. By leveraging GFM's controlled nondeterminism, we present techniques for solving MDPs with LTLf+/PPLTL+ objectives that maintain the compositional DFA-based approach of [Aminof *et al.*, 2024]. Instead of using Emerson-Lei automata, our construction obtains simple LDBAs from the DFAs of LTLf/PPLTL components corresponding to Manna and Pnueli's four classes, then composes them while preserving the "good for MDPs" property. The result is a GFM Büchi automaton that can be used to solve MDPs in a standard way [Baier and Katoen, 2008]. This gives us a simple technique that not only is sound, complete, and computationally optimal, but is both implementation-friendly and well-suited for straightforward symbolic implementation.

2 Preliminaries

In the whole paper, we will fix a set of atomic propositions AP. We denote by $\Sigma = 2^{\mathsf{AP}}$ the set of interpretations over AP; Σ is also called the *alphabet* set. Let Σ^* and Σ^ω denote the set of all finite and infinite sequences, respectively. The empty sequence is denoted as ϵ and we index a sequence $u = a_0 a_1 \cdots a_n \cdots$ from 0. Moreover, we let $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. We denote by $w[i \cdots j]$ the fragment that starts at position i and ends (inclusively) at position j. Particularly, $\epsilon = w[i \cdots j]$ for all w if $j < i$. We denote by $|w|$ the number of letters in w if w is a finite sequence and ∞ otherwise. For a finite or infinite sequence w, $w[0 \cdots i]$ is said to be a *prefix* of w if $0 \leq i < |w|$. A *trace* is a *non-empty* finite or infinite sequence of letters in Σ; Specially, ϵ is *not* a trace.

2.1 LTLf+ and PPLTL+ over Infinite Traces

LTLf+ and PPLTL+ have been derived from LTLf and PPLTL, respectively [Aminof *et al.*, 2024]. The syntax of an LTLf formula [Baier and McIlraith, 2006; De Giacomo and Vardi, 2013] over a finite set of propositions AP is defined as $\phi ::= a \in \mathsf{AP} \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi \mid \mathsf{F}\phi \mid \mathsf{G}\phi$. Here X (strong Next), U (Until), F (or $\diamond$) (Finally/Eventually), and G (or $\square$) (Globally/Always) are temporal operators. The syntax of Pure Past LTL over finite traces (PPLTL) is given as $\phi ::= a \in \mathsf{AP} \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{Y}\phi \mid \phi \mathsf{S} \phi$. Here Y ("Yesterday") and S ("Since") are the past operators, analogues of "Next" and "Until", respectively, but in the past.

Although LTLf and PPLTL have the same expressive power, translating LTLf to DFAs requires a doubly exponential blow-up [De Giacomo and Vardi, 2013], while translating PPLTL to DFAs requires only a singly exponential blow-up [De Giacomo *et al.*, 2020]. We refer interested readers to [De Giacomo and Vardi, 2013] and [De Giacomo *et al.*, 2020] for the semantics of LTLf and PPLTL, respectively. The language of an LTLf/PPLTL formula ϕ, denoted $[\phi]$, is the set of *finite traces* over 2^{AP} that satisfy ϕ.

The syntax of LTLf+ (resp. PPLTL+) is given by the following grammar:

$$\Psi ::= \forall\phi \mid \exists\phi \mid \forall\exists\phi \mid \exists\forall\phi \mid \Psi \vee \Psi \mid \Psi \wedge \Psi \mid \neg\Psi$$

where ϕ are *finite-trace* formulas in LTLf/PPLTL over AP.

Let $w \in \Sigma^\omega$ be an infinite trace and ϕ an LTLf/PPLTL formula. We use $\models_+$ for *non-empty* finite traces and $\models$ for infinite traces. The semantics of an LTLf+/PPLTL+ formula is defined by quantifying over the prefixes of infinite traces:

- $w \models \forall\phi$ means that, for all $i \geq 0$, $w[0 \cdots i] \models_+ \phi$.
- $w \models \exists\phi$ means that there exists an integer $i \geq 0$ such that $w[0 \cdots i] \models_+ \phi$.
- $w \models \forall\exists\phi$ means that, for every $i \geq 0$, there exists an integer $j \geq i$ such that $w[0 \cdots j] \models_+ \phi$.
- $w \models \exists\forall\phi$ means that there exists an integer $i \geq 0$ such that, for all integer $j \geq i$, $w[0 \cdots j] \models_+ \phi$.

Similarly, we denote by $[\Psi]$ the set of *infinite traces* satisfying the LTLf+/PPLTL+ formula Ψ. It has been shown in [Aminof *et al.*, 2024] that LTLf+, PPLTL+, and LTL have the same expressive power.

2.2 Markov Decision Processes

Following [Baier and Katoen, 2008], a Markov decision process (MDP) $\mathcal{M}$ is a tuple $(S, \mathrm{Act}, \mathsf{P}, s_0, L)$ with a finite set of states S, a set of actions Act, a transition probability function $\mathsf{P} : S \times \mathrm{Act} \times S \to [0, 1]$, an initial state $s_0 \in S$ and a labelling function $L : S \to 2^{\mathsf{AP}}$ that labels a state with a set of propositions that hold in that state. A path ξ of $\mathcal{M}$ is a finite or infinite sequence of alternating states and actions $\xi = s_0 a_0 s_1 a_1 \cdots$, ending with a state if finite, such that for all $i \geq 0$, $a_i \in \mathrm{Act}(s_i)$ and $\mathsf{P}(s_i, a_i, s_{i+1}) > 0$. The sequence $L(\xi) = L(s_0)L(s_1), \cdots$ over AP is called the *trace* induced by the path ξ over $\mathcal{M}$.

Denote by FPaths and IPaths the set of all finite and infinite paths of $\mathcal{M}$, respectively. A strategy σ of $\mathcal{M}$ is a function $\sigma : \mathrm{FPaths} \to \mathsf{Distr}(\mathrm{Act})$ such that, for each $\xi \in \mathrm{FPaths}$, $\sigma(\xi) \in \mathsf{Distr}(\mathrm{Act}(\mathsf{lst}(\xi)))$, where $\mathsf{lst}(\xi)$ is the last state of the finite path ξ and $\mathsf{Distr}(\mathrm{Act})$ denotes the set of all possible distributions over Act. Let $\Omega^{\mathcal{M}}_\sigma(s)$ denote the subset of (in)finite paths of $\mathcal{M}$ that correspond to strategy σ and initial state s_0.

A strategy σ of $\mathcal{M}$ is able to resolve the nondeterminism of an MDP and induces a Markov chain (MC) $\mathcal{M}^\sigma = (S^+, \mathsf{P}_\sigma, \mathsf{AP}, L')$ where for $u = s_0 \cdots s_n \in S^+$, $\mathsf{P}_\sigma(u, u \cdot s_{n+1}) = \mathsf{P}(s_n, \sigma(u), s_{n+1})$ and $L'(u) = L(s_n)$.

A *sub-MDP* of $\mathcal{M}$ is an MDP $\mathcal{M}' = (S', \mathrm{Act}', \mathsf{P}', L)$ where $S' \subseteq S, \mathrm{Act}' \subseteq \mathrm{Act}$ is such that for every $s \in S'$, $\mathrm{Act}'(s) \subseteq \mathrm{Act}(s)$, and P' and L' are obtained from P and L, respectively, when restricted to S' and Act'. In particular, $\mathcal{M}'$ is closed under probabilistic transitions, i.e., for all $s \in S'$ and $a \in \mathrm{Act}'$ we have that $\mathsf{P}'(s, a, s') > 0$ implies that $s' \in S'$. An *end-component* (EC) of an MDP $\mathcal{M}$ is a sub-MDP $\mathcal{M}'$ of

$\mathcal{M}$ such that its underlying graph is strongly connected and it has no outgoing transitions. A maximal end-component (MEC) is an EC $\mathcal{E} = (E, \text{Act}', \mathsf{P}', L)$ such that there is no other EC $\mathcal{E} = (E', \text{Act}'', \mathsf{P}'', L)$ such that $E \subset E'$. An MEC E that cannot reach states outside E is called a *leaf* component.

Theorem 1 ([de Alfaro, 1997; Baier and Katoen, 2008])**.** *Once an end-component E of an MDP is entered, there is a strategy that visits every state-action combination in E with probability 1 and stays in E forever. Moreover, for every strategy the union of the end-components is visited with probability 1. An infinite path of an MC $\mathcal{M}$ almost surely (with probability 1) will enter a leaf component.*

2.3 Automata

A (nondeterministic) transition system (TS) is a tuple $\mathcal{T} = (Q, q_0, \delta)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma \to 2^Q$ is a transition function. We also lift δ to sets as $\delta(S, a) := \bigcup_{q \in S} \delta(q, a)$. A *deterministic* TS is such that if, for each $q \in Q$ and $a \in \Sigma$, $|\delta(q, a)| \leq 1$.

An automaton $\mathcal{A}$ is defined as a tuple $(\mathcal{T}, \alpha)$, where $\mathcal{T}$ is a TS and α is an acceptance condition. A *finite run* of $\mathcal{A}$ on a finite word u of length $n \geq 0$ is a sequence of states $\rho = q_0 q_1 \cdots q_n \in Q^+$ such that, for every $0 \leq i < n$, $q_{i+1} \in \delta(q_i, u[i])$, where $u[i]$ indicates the letter of u in position i.

For finite words, we consider finite automata with deterministic TS, known as deterministic finite automata (DFA), where $\alpha = F \subseteq Q$ is a set of *final* states. A finite word u is accepted by the DFA $\mathcal{A}$ if its run $q_0 \cdots q_n$ ends in a final state $q_n \in F$. For an infinite word w, a *run* of $\mathcal{A}$ on w is an infinite sequence of states $\rho = q_0 q_1 q_2 \cdots$ such that, for every $i \geq 0$, $q_{i+1} \in \delta(q_i, w[i])$. Let $\inf(\rho)$ be the set of states that occur infinitely often in the run ρ. We consider the following acceptance conditions for automata on infinite words:

Büchi/co-Büchi. $\alpha = F \subseteq Q$ is a set of *accepting* (*rejecting*, resp.) states for Büchi (co-Büchi, resp.). A run ρ satisfies the Büchi (co-Büchi, resp.) acceptance condition α if $\inf(\rho) \cap F \neq \emptyset$ ($\inf(\rho) \cap F = \emptyset$, resp.).

Rabin. $\alpha = \bigcup_{i=1}^{k} \{(B_i, G_i)\}$ is such that $B_i \subseteq Q$ and $G_i \subseteq Q$ for all $1 \leq i \leq k$. A run ρ satisfies α if there is some $j \in [1, k]$ such that $\inf(\rho) \cap G_j \neq \emptyset$ and $\inf(\rho) \cap B_i = \emptyset$.

A run is *accepting* if it satisfies the condition α; A word $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ if there is an accepting run ρ of $\mathcal{A}$ over w. We use three letter acronyms in $\{D, N\} \times \{F, B, C, R\} \times \{A\}$ to denote automata types where the first letter stands for the TS mode, the second for the acceptance type and the third for automaton. For instance, DBA stands for deterministic Büchi automaton. An NBA is called a *limit deterministic* Büchi automaton (LDBA) if its TS becomes deterministic after seeing accepting states in a run. We assume that all automata are *complete*, i.e., for each state $s \in Q$ and letter $a \in \Sigma$, $|\delta(s, a)| \geq 1$.

We denote by $\mathcal{L}_*(\mathcal{A})$ the set of finite words accepted by a DFA $\mathcal{A}$ or the language of $\mathcal{A}$. Similarly, we denote by $\mathcal{L}(\mathcal{A})$ the *ω-language* recognized by an ω-automaton $\mathcal{A}$, i.e., the set of ω-words accepted by $\mathcal{A}$.

3 Classic MDP Synthesis Approach

In non-probabilistic scenarios, such as reactive synthesis and planning, deterministic ω-automata have to be constructed, e.g. [Aminof *et al.*, 2024]. In contrast, in the probabilistic setting, there can be a controlled form of nondeterminism called *good for MDPs (GFM)* due to the effect of probabilities [Hahn *et al.*, 2020; Schewe *et al.*, 2023]. More precisely, to synthesise a strategy σ for an MDP $\mathcal{M}$ that maximises the satisfaction probability of a given temporal objective Ψ, we do the following steps: first, we construct a GFM automaton $\mathcal{A}$ that recognises $[\Psi]$, then build the product MDP $\mathcal{M} \times \mathcal{A}$, and finally synthesise a strategy σ on $\mathcal{M} \times \mathcal{A}$ that maximises the probability of reaching accepting MECs. Our MDP synthesis algorithm with LTLf+ and PPLTL+ objectives follows the same methodology; our main contribution is a construction of LTLf+/PPLTL+ to GFM automata.

To make our presentation more general, we will assume that we are given a (possibly nondeterministic) ω-automaton $\mathcal{A} = (Q, \delta, q_0, \alpha)$ as specification and an MDP $\mathcal{M} = (S, \text{Act}, \mathsf{P}, s_0, L)$. To find an optimal strategy σ, we define the semantic satisfaction probability of the induced MC $\mathcal{M}^\sigma$ for $\mathcal{L}(\mathcal{A})$ as $\mathbb{P}_{\mathcal{M}^\sigma}(\mathcal{L}(\mathcal{A})) = \mathbb{P}\{\xi \in \Omega_\sigma^{\mathcal{M}}(s_0) : L(\xi) \in \mathcal{L}(\mathcal{A})\}$.

For an MDP $\mathcal{M}$, we define the maximal semantic satisfaction probability as $\mathsf{Psem}(\mathcal{M}, \mathcal{A}) = \sup_\sigma \mathbb{P}_{\mathcal{M}^\sigma}(\mathcal{L}(\mathcal{A}))$. Clearly, for two language-equivalent automata $\mathcal{A}$ and $\mathcal{B}$, it holds that $\mathsf{Psem}(\mathcal{M}, \mathcal{A}) = \mathsf{Psem}(\mathcal{M}, \mathcal{B})$.

Product MDP. As aforementioned, we find the strategy that obtains $\mathsf{Psem}(\mathcal{M}, \mathcal{A})$ by building $\mathcal{M} \times \mathcal{A}$, formally defined as $\mathcal{M} \times \mathcal{A} = (S \times Q, \text{Act} \times Q, \mathsf{P}^\times, \langle s_0, q_0 \rangle, L^\times, \alpha^\times)$ augmented with the acceptance condition $\alpha^\times$ where

- $\mathsf{P}^\times : (S \times Q) \times (\text{Act} \times Q) \times (S \times Q) \to [0, 1]$ such that $\mathsf{P}^\times(\langle s, q \rangle, \langle a, q' \rangle, \langle s', q' \rangle) = \mathsf{P}(s, a, s')$ if $\mathsf{P}(s, a, s') > 0$ and $q' \in \delta(q, L(s))$,
- $L^\times(\langle s, q \rangle) = L(s)$ for a state $\langle s, q \rangle \in S \times Q$, and
- For Büchi/co-Büchi, $\alpha^\times = F^\times = \{\langle s, q \rangle \in S \times Q : q \in F\}$, while for Rabin, $B_i^\times = \{\langle s, q \rangle : q \in B_i\}$ and $G_i^\times = \{\langle s, q \rangle : q \in G_i\}$ for $i \in [1, k]$.

Intuitively, $\mathcal{M} \times \mathcal{A}$ resolves the nondeterminism in $\mathcal{A}$ by making each successor an explicit action. Then, we can generate traces in $\mathcal{L}(\mathcal{A})$ by enforcing the acceptance condition α. Now we define the maximal *syntactic* satisfaction probability:

$$\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \sup_\sigma \mathbb{P}\{\xi \in \Omega_\sigma^{\mathcal{M} \times \mathcal{A}}(\langle s_0, q_0 \rangle) : \xi \text{ is accepting}\}.$$

Clearly, $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) \leq \mathsf{Psem}(\mathcal{M}, \mathcal{A})$ because accepting runs ξ only occur on accepting words. Moreover, $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \mathsf{Psem}(\mathcal{M}, \mathcal{A})$ if $\mathcal{A}$ is deterministic.

An EC containing an infinite run that visits all states and transitions yet satisfies $\alpha^\times$ is said to be accepting. According to Theorem 1, all state-action pairs in an EC can be visited with probability 1. Since an accepting run of $\mathcal{M} \times \mathcal{A}$ must eventually enter an accepting MEC [Baier and Katoen, 2008], the syntactic satisfaction probability can be formalised as:

$$\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \sup_\sigma \mathbb{P}_{(\mathcal{M} \times \mathcal{A})^\sigma}(\diamond X)$$

where X is the set of states of the accepting MECs (AMECs) in $\mathcal{M} \times \mathcal{A}$. GFM automata are the ω-automata, whose nondeterminism can be correctly resolved by the strategy. Formally, An ω*-automaton* $\mathcal{A}$ *is GFM* if, for all finite MDPs $\mathcal{M}$, $\mathsf{Psem}(\mathcal{M}, \mathcal{A}) = \mathsf{Psyn}(\mathcal{M}, \mathcal{A})$ holds [Hahn *et al.*, 2020].

Typical GFM automata include: i) deterministic automata, ii) good-for-games automata [Henzinger and Piterman, 2006] that have a strategy to produce an accepting run for every accepting word, and iii) LDBAs that satisfy certain conditions [Hahn *et al.*, 2015; Sickert *et al.*, 2016].

To use GFM automata in synthesis, we describe a game-theoretic approach to decide what automata are GFM.

AEC-simulation game. While determining the GFMness of an NBA is PSPACE-hard [Schewe *et al.*, 2023], we can use the two-player *accepting end-component simulation* (AEC simulation) game [Hahn *et al.*, 2020] between Spoiler and Duplicator to prove that our constructed automata are GFM. Specially, given a GFM automaton $\mathcal{A}$ and an automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$, if Duplicator wins the AEC-simulation game, $\mathcal{B}$ is also GFM.

In the game, Spoiler places a pebble on the initial state of $\mathcal{A}$, and Duplicator responds by placing a pebble on the initial state of $\mathcal{B}$. The players moves alternately: Spoiler chooses a letter and transition in $\mathcal{A}$, and Duplicator chooses the corresponding transition over the same letter in $\mathcal{B}$. Unlike classic simulation games, Spoiler can, once during the game, make an AEC claim, that she has reached an AEC and provide all transition sequences that will henceforth occur infinitely often in $\mathcal{A}$. Those transitions cannot be updated afterwards. The game continues with both players producing infinite runs in their respective automata. Duplicator wins if: (1) Spoiler never makes an AEC claim, (2) the run constructed in $\mathcal{B}$ is accepting, (3) the run constructed in $\mathcal{A}$ does not comply with the AEC claim, or (4) the run constructed in $\mathcal{A}$ is not accepting. We say that $\mathcal{B}$ AEC-simulates $\mathcal{A}$, if Duplicator wins.

Theorem 2 ([Hahn *et al.*, 2020])**.** *If $\mathcal{A}$ is GFM, $\mathcal{B}$ AEC-simulates $\mathcal{A}$ and $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$, then $\mathcal{B}$ is also GFM.*

The intuition behind Theorem 2 is that, for any MDP $\mathcal{M}$, by Theorem 1, an accepting run of $\mathcal{M} \times \mathcal{A}$ eventually enters an AMEC with probability 1, and the transitions infinitely visited in that AMEC are fixed. When Spoiler makes the AEC claim as the run enters the AMEC, Duplicator can select an accepting run in $\mathcal{M} \times \mathcal{B}$ based on the fixed list of finite traces. Hence, $\mathsf{Psyn}(\mathcal{M}, \mathcal{B}) \geq \mathsf{Psyn}(\mathcal{M}, \mathcal{A})$. Since $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$ and $\mathcal{A}$ is GFM, we have $\mathsf{Psyn}(\mathcal{M}, \mathcal{B}) \geq \mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \mathsf{Psem}(\mathcal{M}, \mathcal{A}) \geq \mathsf{Psem}(\mathcal{M}, \mathcal{B})$. The key idea is that, once an AMEC is entered, the list of infinitely visited transitions is fixed due to probability, unlike in the usual simulation game.

GFM automata are more succinct than deterministic automata [Sickert *et al.*, 2016; Schewe and Tang, 2023]. This then means that we can obtain smaller product MDP $\mathcal{M} \times \mathcal{A}$ with GFM automata and thus a smaller strategy σ since σ uses the states in $\mathcal{M} \times \mathcal{A}$ as memory.

We now give a useful observation to prove that our constructed automata from LTLf+/PPLTL+ are GFM. It basically says that language equivalent GFM automata can AEC-simulate each other when an MC is given. Our proof idea is simple: we just use the optimal strategy σ of $\mathcal{M} \times \mathcal{B}$ that obtains $\mathsf{Psem}(\mathcal{M}, \mathcal{B})$ for Duplicator to play against Spoiler. In this way, Duplicator can always win the AEC-simulation game because $\mathsf{Psem}(\mathcal{M}, \mathcal{A}) = \mathsf{Psem}(\mathcal{M}, \mathcal{B})$.

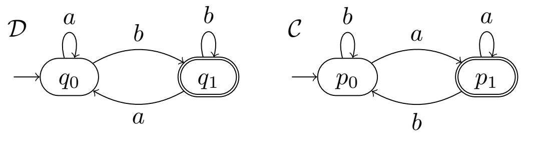

Figure 1: The example DFA $\mathcal{D}$ for LTLf formula $\phi := \mathsf{F}(\mathsf{last} \wedge \text{good})$ that accepts a finite trace in which the proposition good holds at the last position, and the DFA $\mathcal{C}$ for the language $[\neg\phi]$. Here last indicates the last position of a finite trace, i.e., $\mathsf{last} := \neg(\mathsf{X}\mathsf{true})$ and $\Sigma := 2^{\mathsf{AP}} = \{a := \neg\text{good}, b := \text{good}\}$.

Theorem 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be two GFM automata such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$. For any MC $\mathcal{M}$, there is a strategy σ for $\mathcal{M} \times \mathcal{B}$ to AEC-simulate $\mathcal{M} \times \mathcal{A}$.*

With these preparations, we present our synthesis approach: Section 4 covers LTLf+/PPLTL+ to GFM automata, and Section 4.3 outlines our synthesis method.

4 LTLf+/PPLTL+ to GFM Büchi Automata

For a given LTLf+/PPLTL+ formula Ψ, we first construct DFAs for its LTLf/PPLTL subformulas, followed by automata operations to derive the final GFM automaton. Existing constructions of GFM LDBAs [Hahn *et al.*, 2015; Sickert *et al.*, 2016] and GFM NBAs [Hahn *et al.*, 2020] from LTL rely on formula unfolding and construct an explicit state automaton without minimisation.[1] In contrast, our approach has two key advantages: First, it leverages efficient DFA-based techniques including minimisation at every intermediate step; Second, it employs a compositional methodology where the automata for subformulas are constructed and optimised independently before being combined, rather than working with the formula as a whole. Moreover our construction can exploit symbolic techniques. These aspects allow us to better handle complex specifications by controlling state space growth throughout the process. This approach applies to PPLTL [De Giacomo *et al.*, 2020].

We assume that we have a construction at hand to efficiently build the DFA for a given LTLf/PPLTL formula ϕ. More precisely, we denote by $\mathsf{DFA}(\phi)$ the DFA constructed for ϕ. Note that, $\mathsf{DFA}(\phi)$ does not accept the empty sequence ϵ by definition[2]. Figure 1 shows a DFA for the LTLf formula $\phi := \mathsf{F}(\mathsf{last} \wedge \text{good})$, alongside a DFA for the language $[\neg\phi]$.

4.1 Construction for $\exists\phi, \forall\phi, \forall\exists\phi$, and $\exists\forall\phi$

Next we describe the GFM automata constructions of the formulas $\exists\phi, \forall\phi, \forall\exists\phi$ and $\exists\forall\phi$, which we call *leaf* formulas.

$\exists\phi$. We first present the construction for $\exists\phi$ below.

1. First, let $\mathcal{D} = (\Sigma, Q, \iota, \delta, F)$ be $\mathsf{DFA}(\phi)$ such that $\mathcal{L}_*(\mathcal{D}) = [\phi]$.

[1]For a discussion on minimisation on automata on infinite words, see [Schewe, 2010].

[2]In some literature, $\epsilon \in [\phi]$ is allowed. In this situation, we must make sure that $\mathcal{L}_*(\mathsf{DFA}(\phi)) = \Sigma^+ \cap [\phi]$ since the evaluation of the satisfaction for the LTLf+ and PPLTL+ formulas quantifies over nonempty finite traces of the infinite trace.

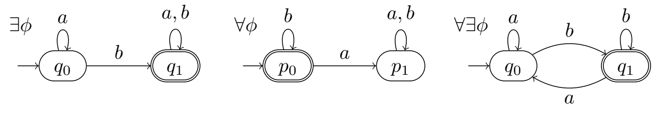

Figure 2: The corresponding Büchi automata constructed by our algorithm for $\exists\phi, \forall\phi$ and $\forall\exists\phi$ where ϕ is as defined in Figure 1.

2. Second, make all final states in F sink final states, i.e., $\delta(s, a) = s$ for each $s \in F$ and $a \in \Sigma$, obtaining the automaton $\mathcal{C} = (Q, \iota', \delta', F')$. $\mathcal{C}$ remains deterministic.
3. Finally, read it as DBA $\mathcal{B} = (Q, \iota', \delta', F')$.

Theorem 4. $\mathcal{L}(\mathcal{B}) = [\exists\phi]$.

By making final states sink, every word accepted in $\mathcal{B}$ must have a prefix belonging to $[\phi]$. Hence, the theorem follows.

$\forall\phi$**.** Now we introduce the construction for $\forall\phi$.

1. First, let $\mathcal{C} = (Q, \iota, \delta, F)$ be $\mathsf{DFA}(\neg\phi)$ such that $\mathcal{L}_*(\mathcal{C}) = [\neg\phi]$.
2. Then, make all final states in $\mathcal{C}$ as sink final states, remove unreachable states and obtain the DFA $\mathcal{C}' = (Q', \iota, \delta', F')$
3. Finally, reverse the set of final states and read it as Büchi automaton $\mathcal{B} = (Q', \iota, \delta', Q' \setminus F')$.

Theorem 5. $\mathcal{L}(\mathcal{B}) = [\forall\phi]$.

By making final states of $\mathcal{C}$ sink states, every accepting run over a word w in $\mathcal{B}$ does not visit those sink final states in $\mathcal{C}$, which then entails that no prefixes of w belong to $[\neg\phi]$. That is, all prefixes of w belong to ϕ. Then the theorem follows. Our construction for $\forall\phi$ is quite similar to the one in [Bansal *et al.*, 2023].

$\forall\exists\phi$**.** The construction for $\forall\exists\phi$ is simple and given below.

1. First, let $\mathcal{D} = (Q, \iota, \delta, F)$ be the DFA $\mathsf{DFA}(\phi)$.
2. Then, read $\mathcal{D}$ as Büchi automaton $\mathcal{B} = (Q, \iota, \delta, F)$.

Theorem 6. $\mathcal{L}(\mathcal{B}) = [\forall\exists\phi]$.

Theorem 6 clearly holds since every accepting run in $\mathcal{B}$ has a finite prefix run that is an accepting run in $\mathcal{D}$. Figure 2 shows the Büchi automata constructed for $\exists\phi, \forall\phi$ and $\forall\exists\phi$.

$\exists\forall\phi$**.** The construction for $\exists\forall\phi$ is more involved and the flowchart is depicted below. The idea is to first build the DCA $\mathcal{A}$ for $\exists\forall\phi$, which is also the DBA for $\forall\exists\neg\phi$ and then convert $\mathcal{A}$ to the desired LDBA $\mathcal{B}$ accepting $[\exists\forall\phi]$. In detail:

$$\begin{tikzcd}
\text{DFA } \mathcal{D} \arrow[r, "\text{complement}"] & \text{DFA } \mathcal{C} \arrow[r, "\text{read as}"] & \text{DCA } \mathcal{A} \arrow[dl, "\text{convert}"] \\
\text{LDBA } \mathcal{B} & \text{LDBA } \mathcal{B}' \arrow[l, "\text{complete}"'] &
\end{tikzcd}$$

1. First, complement $\mathcal{D}$ by reversing the set of final states, and obtain the DFA $\mathcal{C} = (Q, \iota, \delta, Q \setminus F)$ for $\neg\phi$.
2. Second, read $\mathcal{C}$ as a co-Büchi automaton $\mathcal{A} = (Q, \iota, \delta, Q \setminus F)$. If we treat $\mathcal{C}$ as a Büchi automaton $\mathcal{G}$, by Theorem 6, $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\forall\exists\neg\phi)$. Since $\mathcal{A}$ is dual to $\mathcal{G}$, we immediately have $\mathcal{L}(\mathcal{A}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{G}) = \mathcal{L}(\exists\forall\phi)$.
3. Third, convert $\mathcal{A}$ to a LDBA $\mathcal{B}' = (Q \times \{0, 1\}, \langle\iota, 0\rangle, \delta'', F \times \{1\})$ where $\delta'' = \delta_0 \uplus \delta_j \uplus \delta_1$ is defined as follows:
   - $\langle q', 0\rangle = \delta_0(\langle q, 0\rangle, a)$ for all $q \in Q, q' \in Q$ and $a \in \Sigma$ with $\delta(q, a) = q'$,
   - $\langle q', 1\rangle = \delta_1(\langle q, 1\rangle, a)$ for all $q \in F, q' \in F$ and $a \in \Sigma$ with $\delta(q, a) = q'$,
   - $\langle q', 1\rangle = \delta_j(\langle q, 0\rangle, a)$ for all $q \in Q, q' \in F$ and $a \in \Sigma$ with $\delta(q, a) = q'$
4. Finally, complete the LDBA $\mathcal{B}'$ and obtain the result $\mathcal{B} = (Q', \iota, \delta', F' = F \times \{1\})$.

An accepting run of $\mathcal{G}$ must visit $Q \setminus F$ infinitely often; equivalently, an accepting run of $\mathcal{A}$ must visit only F-states from some point on. The intuition behind the NBA $\mathcal{B}'$ is that it has to guess that point via the transition δ_j. Thus, before that point, $\mathcal{B}'$ stays within the component $Q \times \{0\}$, and once the run of $\mathcal{B}'$ has entered the component $F \times \{1\}$ via δ_j, it visits only states in $F \times \{1\}$ from that moment forward.

Theorem 7. $\mathcal{L}(\mathcal{B}) = [\exists\forall\phi]$.

While the DBAs for $\forall\phi$ and $\exists\phi$ can be minimized in polynomial time [Löding, 2001], minimizing the ones for $\forall\exists\phi$ and $\exists\forall\phi$ can be NP-complete [Schewe, 2010]. Nonetheless, we can apply cheaper reduction operations such as *extreme minimisation* [Badr, 2009] to $\mathcal{B}$.While the construction gives a LDBA instead of DBAs, the resulting LDBA is also GFM.

Theorem 8. *The automata $\mathcal{B}$ for each of $\exists\phi, \forall\phi, \forall\exists\phi$, and $\exists\forall\phi$ are GFM.*

Proof. In fact we have seen that the automaton $\mathcal{B}$ for $\exists\phi, \forall\phi, \forall\exists\phi$ is a DBA so trivially GFM. We prove the result for the automaton $\mathcal{B}$ for $\exists\forall\phi$ by creating an AEC-simulation game between Spoiler and Duplicator. Spoiler will play on DCA $\mathcal{A}$ and Duplicator will play on $\mathcal{B}$. First, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B}) = [\exists\forall\phi]$. Moreover, $\mathcal{A}$ is a GFM automaton because $\mathcal{A}$ is deterministic. Therefore, if we can prove that $\mathcal{B}$ AEC-simulates $\mathcal{A}$, then $\mathcal{B}$ is also GFM according to theorem 2.

Now we provide a winning strategy for Duplicator on $\mathcal{B}$ in the AEC-simulation game. Before Spoiler makes the AEC claim, Duplicator will take transitions within $Q \times \{0\}$ via δ_0. Once the AEC claim is made by Spoiler, the Duplicator will transition to $F \times \{1\}$ via δ_j at the earliest point. Afterwards, Duplicator takes transitions within $F \times \{1\}$ via δ_1. The only situation where Spoiler wins is that the run ρ in $\mathcal{A}$ over the chosen word w is accepting. That is, ρ cannot visit $Q \setminus F$ states any more after the AEC claim since the list of finite traces visited infinitely often is fixed. This then entails that the run $\widehat{\rho}$ constructed by Duplicator will also be accepting in $\mathcal{B}$ since the projection on Q of $\widehat{\rho}$ is exactly ρ. It follows that Duplicator wins the game and $\mathcal{B}$ AEC-simulates $\mathcal{A}$. Hence, $\mathcal{B}$ is also GFM according to Theorem 2. □

Following example, illustrates why the constructed LDBA $\mathcal{B}$ for $\exists\forall\phi$ is GFM.

Example 1. *Consider the DFA $\mathcal{D}$ for ϕ on the upper left of Figure 3. The DCA $\mathcal{A}$ (or DFA $\mathcal{C}$) with rejecting states $\{q_0\}$ is shown on the lower left, and the complete LDBA $\mathcal{B}$ is on the right (LDBA $\mathcal{B}'$ is within the dashed box). The*

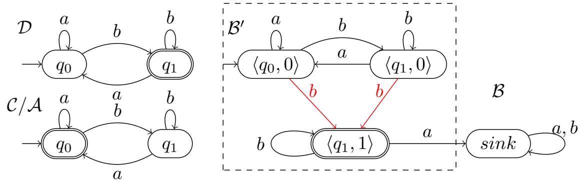

Figure 3: The DFA $\mathcal{D}$ for ϕ is on upper left, the DCA $\mathcal{A}$ (or DFA $\mathcal{C}$) with the set of rejecting states $\{q_0\}$ is on lower left, $\mathcal{A}$ has language $\{a, b\}^* \cdot b^\omega$ and the LDBA $\mathcal{B}'$ is the part within the dashed box and $\mathcal{B}$ is the complete LDBA where $\{\langle q_1, 1\rangle\}$ is the sole accepting state marked with double rounded boxes and the jump transitions in δ_j are drawn in red colour.

sole accepting state $\{\langle q_1, 1\rangle\}$ is marked with double rounded boxes, and jump transitions in δ_j are in red. A winning strategy for the Duplicator allows $\mathcal{B}$ to AEC-simulate $\mathcal{A}$. Let δ and δ' be the transition functions of $\mathcal{A}$ and $\mathcal{B}$, respectively. The strategy σ works as follows: (1) Before the AEC-claim, $\sigma(u) = \delta'(\langle q_0, 0\rangle, u)$; (2) At the AEC claim, σ takes the jump transition upon reading b as soon as possible; (3) Afterwards, σ uses δ' for successors. The Spoiler can win only by making an AEC claim and forcing $\mathcal{A}$ to stay in q_1 forever. In this case, σ ensures an accepting run.

4.2 Boolean Combinations of GFM Automata

Now, we show that GFM automata are closed under union and intersection. Let $\mathcal{A}_0$ and $\mathcal{A}_1$ be two GFM Büchi automata. First we introduce the union operation for $\mathcal{A}_0$ and $\mathcal{A}_1$.

Proposition 1. *Given two Büchi automata $\mathcal{A}_0 = (Q_0, \iota_0, \delta_0, F_0)$ and $\mathcal{A}_1 = (Q_1, \iota_1, \delta_1, F_1)$, let $\mathcal{A} = (Q, \iota, \delta, F)$ be the Büchi automaton where $Q = Q_0 \times Q_1$, $\iota = (\iota_0, \iota_1)$, $\delta(\langle q_0, q_1\rangle, a) = \delta_0(q_0, a) \times \delta_1(q_1, a)$, and $F = Q_0 \times F_1 \cup F_0 \times Q_1$. Then, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_0) \cup \mathcal{L}(\mathcal{A}_1)$ with $|Q| = |Q_0| \cdot |Q_1|$.*

This union operation is just a Cartesian product of $\mathcal{A}_0$ and $\mathcal{A}_1$. Let $w \in \Sigma^*$. For the word w, a run ρ_0 of $\mathcal{A}_0$ and a run ρ_1 of $\mathcal{A}_1$ constitute a run $\mathcal{A}$ in the form of $\rho_0 \times \rho_1$. So, if one of the runs is accepting, $\rho_0 \times \rho_1$ is also accepting. We show below that the union automaton is also GFM.

Theorem 9. *If $\mathcal{A}_0$ and $\mathcal{A}_1$ are both GFM, then the union automaton $\mathcal{A}$ is also GFM.*

Proof. Let $\mathcal{M} = (S, \mathrm{Act}, \mathsf{P}, s_0, L)$ be an MDP. Our proof goal is to show that $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \mathsf{Psem}(\mathcal{M}, \mathcal{A})$. Then, we can just prove that $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) \geq \mathsf{Psem}(\mathcal{M}, \mathcal{A})$. We will prove it with the help of equivalent DRAs of $\mathcal{A}_0$ and $\mathcal{A}_1$.

Let $\mathcal{R}_0 = (Q'_0, \delta'_0, \iota'_0, \alpha'_0)$ and $\mathcal{R}_1 = (Q'_1, \delta'_1, \iota'_1, \alpha'_1)$ be two DRAs that are language-equivalent to $\mathcal{A}_0$ and $\mathcal{A}_1$, respectively. Let $\mathcal{R} = \mathcal{R}_0 \times \mathcal{R}_1$ be the union DRA of $\mathcal{R}_0$ and $\mathcal{R}_1$ such that $\mathcal{L}(\mathcal{R}) = \mathcal{L}(\mathcal{R}_0) \cup \mathcal{L}(\mathcal{R}_1)$. Formally, $\mathcal{R}$ is a tuple $(Q' = Q'_0 \times Q'_1, \delta', \iota' = \langle \iota'_0, \iota'_1\rangle, \alpha')$ where $\delta'(\langle q_0, q_1\rangle, a) = \langle \delta'_0(q_0, a), \delta'_1(q_1, a)\rangle$ for each $\langle q_0, q_1\rangle \in Q'$ and $a \in \Sigma$, and $\alpha' = \bigcup_{i=1}^{k_0}\{(B_i \times Q'_1, G_i \times Q'_1) : (B_i, G_i) \in \alpha'_0\} \cup \bigcup_{i=1}^{k_1}\{(Q'_0 \times B_i, Q'_0 \times G_i) : (B_i, G_i) \in \alpha'_1\}$. Let $w \in \Sigma^\omega$, and ρ_0 and ρ_1 are the runs over w in $\mathcal{R}_0$ and $\mathcal{R}_1$, respectively. The run of $\mathcal{R} = \mathcal{R}_0 \times \mathcal{R}_1$ over w is actually the product $\rho_0 \times \rho_1$. Moreover, if $w \in \mathcal{L}(\mathcal{A})$, then the run $\rho_0 \times \rho_1$ satisfies either α'_0 or α'_1, which indicates that $\rho_0 \times \rho_1$ is accepting in $\mathcal{R}$. Then, it follows that $\mathsf{Psem}(\mathcal{M}, \mathcal{R}) = \mathsf{Psem}(\mathcal{M}, \mathcal{A})$ as $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{R})$. As $\mathcal{R}$ is deterministic, we have $\mathsf{Psem}(\mathcal{M}, \mathcal{R}) = \mathsf{Psyn}(\mathcal{M}, \mathcal{R})$. Therefore, we only need to prove that $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) \geq \mathsf{Psyn}(\mathcal{M}, \mathcal{R})$.

Let σ be the *optimal* strategy on $\mathcal{M}$ to obtain the maximal satisfaction probability for $\mathcal{L}(\mathcal{A})$, i.e.,

$$\mathbb{P}_{\mathcal{M}^\sigma}(\mathcal{L}(\mathcal{A})) = \sup_{\sigma'} \mathbb{P}\{\xi \in \Omega^{\mathcal{M}}_{\sigma'}(s_0) : L(\xi) \in \mathcal{L}(\mathcal{A})\}.$$

Note again that here σ is usually not a positional strategy for $\mathcal{M}$ and needs extra memory to store history traces.

We denote by $\mathcal{T}_0$, $\mathcal{T}_1$ and $\mathcal{T}_0 \times \mathcal{T}_1$ the TSes of $\mathcal{R}_0$, $\mathcal{R}_1$, and $\mathcal{R}$ respectively. We now work on the large Markov chain $\mathcal{M}' = \mathcal{M}^\sigma \times \mathcal{T}_0 \times \mathcal{T}_1$ by ignoring the acceptance conditions where $\mathcal{M}^\sigma$ is already an MC. Thus, $\mathcal{M}'$ has only probabilistic choices. Since $\mathcal{R} = \mathcal{R}_0 \times \mathcal{R}_1$ is deterministic, we have that

$$\mathsf{Psem}(\mathcal{M}, \mathcal{A}) = \mathsf{Psyn}(\mathcal{M}, \mathcal{R}) = \mathsf{Psyn}(\mathcal{M}', \mathcal{R}).$$

Let $c \in \{0, 1\}$. We know that $\mathcal{A}_c$ is GFM. According to Theorem 3, there is an optimal strategy σ_c for $\mathcal{M}' \times \mathcal{A}_c$ to AEC-simulate $\mathcal{M}' \times \mathcal{R}_c$. Thus, we construct the optimal strategy σ^* for $\mathcal{M} \times \mathcal{A}$ by building the product of three strategies σ for $\mathcal{M}$, σ_0 and σ_1, which resolves the nondeterminism of $\mathcal{M}$, $\mathcal{A}_0$ and $\mathcal{A}_1$, respectively, independently in $\mathcal{M}' \times \mathcal{A}$ (viewing $\mathcal{A}$ as the cross product of $\mathcal{A}_0$ and $\mathcal{A}_1$). That is, in this case, σ_c works independently from σ_{1-c} on $\mathcal{M}^\sigma \times \mathcal{T}_0 \times \mathcal{T}_1 \times \mathcal{A}_c$ and the state space of $\mathcal{T}_0 \times \mathcal{T}_1$ will be used as extra memory for σ_c in addition to store states from $\mathcal{M}^\sigma$ and $\mathcal{A}_c$.

Then, in the AEC-simulation game, whenever Spoiler produces an accepting run in $\mathcal{M}' \times \mathcal{R}$, we can use σ^* to construct an accepting run for Duplicator in $\mathcal{M}' \times \mathcal{A}_0 \times \mathcal{A}_1 = \mathcal{M}' \times \mathcal{A}$. So, there is a strategy σ^* on $\mathcal{M} \times \mathcal{A}$ to achieve $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \mathsf{Psyn}(\mathcal{M}', \mathcal{A}) \geq \mathsf{Psyn}(\mathcal{M}', \mathcal{R})$. It then follows that $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) \geq \mathsf{Psyn}(\mathcal{M}', \mathcal{R}) = \mathsf{Psem}(\mathcal{M}, \mathcal{A})$. That is, $\mathsf{Psyn}(\mathcal{M}, \mathcal{A}) = \mathsf{Psem}(\mathcal{M}, \mathcal{A})$ for any given MDP $\mathcal{M}$. Therefore, $\mathcal{A}$ is also GFM. □

We illustrate why the union product $\mathcal{A}$ is GFM if both $\mathcal{A}_0$ and $\mathcal{A}_1$ are GFM with an example.

Example 2. *Consider the GFM NBAs $\mathcal{A}_0$ and $\mathcal{A}_1$ on the left of Figure 4, and their union product $\mathcal{A}$ on the right. $\mathcal{A}_0$ accepts $(\{a, b\}^* \cdot b)^\omega$, and $\mathcal{A}_1$ accepts $(\{a, b\}^* \cdot a)^\omega$. Both are GFM as they have strategies to generate an accepting run for every accepting word. The winning strategy stays in the initial state before the AEC-claim and then aims to visit the accepting state repeatedly. The union product $\mathcal{A}$ is also GFM, with a strategy σ enabling the Duplicator to win the AEC-simulation game against any deterministic automaton $\mathcal{D}$. The strategy σ works as follows: Before the AEC claim, σ keeps the run in the initial state $\langle p_0, q_0\rangle$. After the AEC claim, σ exits $\langle p_0, q_0\rangle$ and confines the run to $\{\langle p_0, q_1\rangle, \langle p_1, q_0\rangle\}$. For any accepting infinite word chosen by the Spoiler, σ ensures an accepting run. In fact, σ combines the winning strategies of $\mathcal{A}_0$ and $\mathcal{A}_1$. Hence, $\mathcal{A}$ is GFM.*

The Cartesian product is necessary for $\mathcal{A}$ being GFM. The normal union product only adds an extra initial state to nondeterministically select one of the two automata for the subsequent run, which has only $|Q_1| + |Q_2| + 1$ states. However, the product might produce an automaton that is not GFM [Schewe *et al.*, 2023].

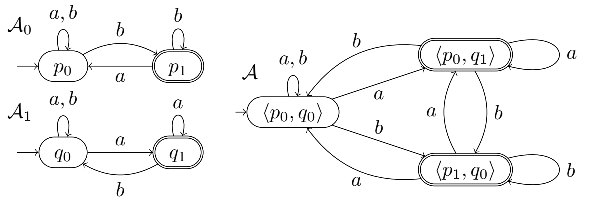

Figure 4: The GFM NBAs $\mathcal{A}_0$ and $\mathcal{A}_1$ are depicted on the left and their union product $\mathcal{A}$ is depicted on the right. The accepting states are marked with double rounded boxes.

Next, we introduce the intersection operation for the GFM automata and show that the result automaton is also GFM.

Proposition 2 ([Kupferman, 2018]). *Given two Büchi automata $\mathcal{A}_0 = (Q_0, \iota_0, \delta_0, F_0)$ and $\mathcal{A}_1 = (Q_1, \iota_1, \delta_1, F_1)$, let $\mathcal{A} = (Q, \iota, \delta, F)$ be the intersection Büchi automaton whose components are defined as follows:*

- $Q = Q_0 \times Q_1 \times \{0, 1\}$*;*
- $\iota = (\iota_0, \iota_1, 0)$*;*
- *For a state $\langle q_0, q_1, c\rangle$ and letter $a \in \Sigma$, we have $\langle q_0', q_1', \textit{next}(q_0, q_1, c)\rangle \in \delta(\langle q_0, q_1, c\rangle, a)$ where $q_0' \in \delta_0(q_0, a), q_1' \in \delta_1(q_1, a)$, next: $Q_0 \times Q_1 \times \{0, 1\} \to \{0, 1\}$ is defined as*
$$\textit{next}(q_0, q_1, c) = \begin{cases} 1 - c & \text{if } q_c \in F_c, \\ c & \text{otherwise;} \end{cases}$$
- $F = F_0 \times Q_1 \times \{0\}$.

Then, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_0) \cap \mathcal{L}(\mathcal{A}_1)$ with $|Q| = 2 \cdot |Q_0| \cdot |Q_1|$.

This intersection construction is fairly standard [Kupferman, 2018]. The intuition is to alternatively look for accepting states from $\mathcal{A}_0$ when c is set to 0 and for accepting states in $\mathcal{A}_1$ when c is 1. In this way, the accepting run of $\mathcal{A}$ must visit accepting states from both $\mathcal{A}_0$ and $\mathcal{A}_1$ infinitely often. The resulting automaton is an LDBA.

Similarly, we show that the GFM automata are also closed under the intersection operation by Theorem 10.

Theorem 10. *If $\mathcal{A}_0$ and $\mathcal{A}_1$ are GFM, then the intersection automaton $\mathcal{A}$ is also GFM.*

In [Hahn *et al.*, 2020], the constructed GFM automata are *slim* in the sense that each state has at most two successors over a letter. The GFM automata we produce can easily be made slim. This is because the only nondeterministic decision we have to make is to guess in the intersection operation when the individual DCAs will henceforth see only accepting states. This decision can obviously be arbitrarily delayed. We can therefore make them round-robin, considering only one LDBA that stems from a DCA at a time. For instance, when we have positive Boolean operations that contain 42 automata from $\exists\forall\phi$ formulas, then we can avoid the outdegree from jumping to 2^{42} at the expense of increasing the state space by a factor of 42 (while retaining the out-degree of 2) to handle the round-robin in the standard way.

By applying the constructions of Büchi automata for leaf formulas and constructions for intersection and union operations on the syntax tree of a LTLf+ or PPLTL+ formula, we can obtain a final Büchi automaton for the formula. Therefore, the following result holds.

Theorem 11. *For an LTLf+/PPLTL+ formula Ψ, our method constructs a GFM Büchi automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = [\Psi]$.*

Let $|\Psi|$ be the length of the formula, i.e., the number of modalities and operations in the formula. Then we have that:

Theorem 12. *For an LTLf+ (respectively, PPLTL+) formula Ψ, the number of states in $\mathcal{A}$ is $2^{2^{\mathcal{O}(|\Psi|)}}$ (respectively $2^{\mathcal{O}(|\Psi|)}$).*

The number of states obtained in Theorem 12 is optimal, in the sense that the worst-case double exponential blow-up (single exponential blow-up) is already unavoidable for LTLf (resp. PPLTL) [Bansal *et al.*, 2023; De Giacomo *et al.*, 2020].

4.3 Returning the Strategy

Given an LTLf+/PPLTL+ formula Ψ and an MDP $\mathcal{M}$, our construction can obtain an optimal strategy for $\mathcal{M}$ to achieve maximal satisfaction probability of Ψ as follows.

First, construct a GFM Büchi automaton $\mathcal{A}$ for Ψ using approaches described in Section 4 with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\Psi)$.

Second, construct the product $\mathcal{M}^\times = \mathcal{M} \times \mathcal{A}$ and compute the list of AMECs $E = \{E_1, \cdots, E_k\}$ in $\mathcal{M}^\times$ using the standard approach described in [Baier and Katoen, 2008].

Finally, synthesise a strategy as follows: In the AMECs, we can select an action for every state that gives the shortest path to the set of accepting states. This shortest path can be multiple but we only need one and the definition of shortest path is clearly well defined. For states outside AMECs, we select an action for every state to reach an AMEC with maximal probability; this is equivalent to computing the strategy for obtaining the maximal reachability probability to AMECs. The resultant strategy, denoted by $\sigma^\times$, is positional on $\mathcal{M}^\times$ for Büchi acceptance condition according to [Mazala, 2001].

Since the Büchi automaton $\mathcal{A}$ for Ψ is GFM we get:

Theorem 13. *The synthesised strategy $\sigma^\times$ for $\mathcal{M}$ is optimal to achieve maximal satisfaction probability of Ψ.*

5 Conclusion

In this paper, we investigated the problem of solving MDPs with LTLf+ and PPLTL+ temporal objectives, showing the effectiveness of these logics for probabilistic planning in MDPs. Our key contribution lies in presenting a provably correct technique to construct GFM Büchi automata for LTLf+ and PPLTL+ formulas, leveraging the compositional advantages of DFA-based methods. Note that our construction is designed to be implementation-friendly and well-suited for a straightforward symbolic implementation. In fact, for PPLTL we can directly construct the symbolic DFA in polynomial time [De Giacomo *et al.*, 2020]. The final Boolean combination of these symbolic Büchi automata, as described in Section 4.2, is at most polynomial in their combined sizes and can be realised through a Cartesian product. As a future work, we aim to implement this approach, employing symbolic techniques, within state-of-the-art tools such as PRISM [Kwiatkowska *et al.*, 2011]. This development will facilitate the practical application of our methods across a range of domains, including AI, robotics, and probabilistic verification.

Acknowledgements

This work has been partially supported by ISCAS Basic Research (Grant Nos. ISCAS-JCZD-202406, ISCAS-JCZD-202302), CAS Project for Young Scientists in Basic Research (Grant No. YSBR-040), ISCAS New Cultivation Project ISCAS-PYFX-202201, the ERC Advanced Grant WhiteMech (No. 834228), the UKRI Erlangen AI Hub on Mathematical and Computational Foundations of AI, and the EPSRC through grants EP/X03688X/1 and EP/X042596/1.

References

[Aminof *et al.*, 2024] Benjamin Aminof, Giuseppe De Giacomo, Sasha Rubin, and Moshe Y. Vardi. LTLf+ and PPLTL+: extending LTLf and PPLTL to infinite traces. *CoRR*, abs/2411.09366, 2024.

[Bacchus and Kabanza, 1998] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Ann. Math. Artif. Intell.*, 22(1-2):5–27, 1998.

[Badr, 2009] Andrew Badr. Hyper-minimization in O(n^2). *Int. J. Found. Comput. Sci.*, 20(4):735–746, 2009.

[Baier and Katoen, 2008] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[Baier and McIlraith, 2006] Jorge A. Baier and Sheila A. McIlraith. Planning with temporally extended goals using heuristic search. In *ICAPS*, pages 342–345. AAAI, 2006.

[Bansal *et al.*, 2023] Suguman Bansal, Yong Li, Lucas M. Tabajara, Moshe Y. Vardi, and Andrew M. Wells. Model checking strategies from synthesis over finite traces. In *ATVA (1)*, volume 14215 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2023.

[Bienvenu *et al.*, 2011] Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith. Specifying and computing preferred plans. *Artif. Intell.*, 175(7-8):1308–1345, 2011.

[Camacho *et al.*, 2019] Alberto Camacho, Meghyn Bienvenu, and Sheila A McIlraith. Towards a unified view of AI planning and reactive synthesis. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 58–67, 2019.

[de Alfaro, 1997] Luca de Alfaro. *Formal verification of probabilistic systems*. PhD thesis, Stanford University, USA, 1997.

[De Giacomo and Rubin, 2018] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTLf and LDLf goals. In *IJCAI*, pages 4729–4735. ijcai.org, 2018.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860. IJCAI/AAAI, 2013.

[De Giacomo and Vardi, 2015] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 1558–1564. AAAI Press, 2015.

[De Giacomo *et al.*, 2020] Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-past linear temporal and dynamic logic on finite traces. In *IJCAI*, pages 4959–4965. ijcai.org, 2020.

[Ehlers *et al.*, 2017] Rüdiger Ehlers, Stéphane Lafortune, Stavros Tripakis, and Moshe Y. Vardi. Supervisory control and reactive synthesis: a comparative introduction. *Discret. Event Dyn. Syst.*, 27(2):209–260, 2017.

[Emerson and Lei, 1987] E. Allen Emerson and Chin-Laung Lei. Modalities for model checking: Branching time logic strikes back. *Sci. Comput. Program.*, 8(3):275–306, 1987.

[Fainekos *et al.*, 2009] Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. *Autom.*, 45(2):343–352, 2009.

[Finkbeiner, 2016] Bernd Finkbeiner. Synthesis of reactive systems. In *Dependable Software Systems Engineering*, volume 45 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, pages 72–98. IOS Press, 2016.

[Hahn *et al.*, 2015] Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. Lazy probabilistic model checking without determinisation. In *CONCUR*, volume 42 of *LIPIcs*, pages 354–367. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[Hahn *et al.*, 2020] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *TACAS (1)*, volume 12078 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2020.

[Hausmann *et al.*, 2024] Daniel Hausmann, Mathieu Lehaut, and Nir Piterman. Symbolic solution of Emerson-Lei games for reactive synthesis. In *FoSSaCS (1)*, volume 14574 of *Lecture Notes in Computer Science*, pages 55–78. Springer, 2024.

[Henzinger and Piterman, 2006] Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 395–410. Springer, 2006.

[Kupferman, 2018] Orna Kupferman. Automata theory and model checking. In *Handbook of Model Checking*, pages 107–151. Springer, 2018.

[Kwiatkowska *et al.*, 2011] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[Löding, 2001] Christof Löding. Efficient minimization of deterministic weak omega-automata. *Inf. Process. Lett.*, 79(3):105–109, 2001.

[Maggi *et al.*, 2011] Fabrizio Maria Maggi, Marco Montali, Michael Westergaard, and Wil M. P. van der Aalst. Monitoring business constraints with linear temporal logic: An approach based on colored automata. In *BPM*, volume

6896 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2011.

[Manna and Pnueli, 1990] Zohar Manna and Amir Pnueli. A hierarchy of temporal properties. In *PODC*, pages 377–410. ACM, 1990.

[Mazala, 2001] René Mazala. Infinite games. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, pages 23–42. Springer, 2001.

[Pnueli and Rosner, 1989] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.

[Schewe and Tang, 2023] Sven Schewe and Qiyi Tang. On the succinctness of Good-for-MDPs automata. *CoRR*, abs/2307.11483, 2023.

[Schewe *et al.*, 2023] Sven Schewe, Qiyi Tang, and Tansholpan Zhanabekova. Deciding what is Good-for-MDPs. In *CONCUR*, volume 279 of *LIPIcs*, pages 35:1–35:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[Schewe, 2010] Sven Schewe. Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete. In *FSTTCS*, volume 8 of *LIPIcs*, pages 400–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[Shao and Kwiatkowska, 2023] Daqian Shao and Marta Kwiatkowska. Sample efficient model-free reinforcement learning from LTL specifications with optimality guarantees. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023.

[Sickert *et al.*, 2016] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Kretínský. Limit-deterministic Büchi automata for linear temporal logic. In *CAV (2)*, volume 9780 of *Lecture Notes in Computer Science*, pages 312–332. Springer, 2016.