Separable GPL: Decidable Model Checking with More Non-Determinism

Andrey Gorlin
Stony Brook University
Department of Computer Science, Stony Brook, N.Y. 11794, USA
agorlin@cs.stonybrook.edu

C. R. Ramakrishnan
Stony Brook University
Department of Computer Science, Stony Brook, N.Y. 11794, USA
cram@cs.stonybrook.edu

Abstract

Generalized Probabilistic Logic (GPL) is a temporal logic, based on the modal mu-calculus, for specifying properties of branching probabilistic systems. We consider GPL over branching systems that also exhibit internal non-determinism under linear-time semantics (which is resolved by schedulers), and focus on the problem of finding the capacity (supremum probability over all schedulers) of a fuzzy formula. Model checking GPL is undecidable, in general, over such systems, and existing GPL model checking algorithms are limited to systems without internal non-determinism, or to checking non-recursive formulae. We define a subclass, called separable GPL, which includes recursive formulae and for which model checking is decidable. A large class of interesting and decidable problems, such as termination of 1-exit Recursive MDPs, reachability of Branching MDPs, and LTL model checking of MDPs, whose decidability has been studied independently, can be reduced to model checking separable GPL. Thus, GPL is widely applicable and, with a suitable extension of its semantics, yields a uniform framework for studying problems involving systems with non-deterministic and probabilistic behaviors.

2012 ACM Subject Classification Theory of computation → Verification by model checking

Keywords and phrases Modal mu-calculus, probabilistic logics, probabilistic systems, branching systems, model checking

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.36

Related Version https://arxiv.org/abs/1604.06118

Funding This work was partially supported by NSF grants IIS-1447549 and CNS-1405641.

Acknowledgements We are grateful to the reviewers for their detailed and insightful comments.

1 Introduction

For finite-state systems, model checking a temporal property can be cast in terms of model checking in the modal μ-calculus, the so-called "assembly language" of temporal logics. A number of temporal logics have been proposed and used for specifying properties of finite-state *probabilistic* systems.

GPL [5] is defined over branching probabilistic systems. In these systems, each state has a set of labeled outgoing transitions; each transition, in turn, specifies a (probabilistic) distribution of target states. Semantically, GPL treats probabilistic choices in the system as

linear-time. GPL is expressive enough to serve as an "assembly language" of a large number of probabilistic temporal logics.

In this paper, we consider GPL over branching systems that can also exhibit internal non-determinism under linear-time semantics. In these systems, the internal non-determinism is resolved by schedulers. We limit our attention to finding the capacity (supremum probability over all schedulers) of fuzzy formulae, as this problem is already sufficiently interesting and challenging; indeed, it is undecidable, in general. Existing GPL model checking algorithms, therefore, were limited either to systems without internal non-determinism [5], or to checking only non-recursive formulae [26].

Contributions and Significance. GPL is expressive enough that a variety of independently-studied verification problems can be cast as model checking branching systems with GPL. In fact, undecidable problems such as termination of multi-exit *Recursive* Markov Decision Processes (Recursive MDPs or RMDPs) can be reduced in linear time to model checking with GPL. We introduce a syntactically-defined subclass, called *separable GPL*, for which model checking is decidable.

We illustrate the expressiveness of separable GPL by considering independently-studied decidable verification problems involving systems that have probabilistic and non-deterministic choice. Examples of such problems include LTL model checking of MDPs [2], reachability in branching MDPs [10], and termination of 1-exit RMDPs [12]. These problems can all be reduced, in linear time, to model checking separable GPL formulae (see Sect. 3).

We describe a procedure for model checking GPL, which either successfully returns the model checking result, or terminates with failure. We also show that the procedure always terminates successfully for separable GPL (see Sect. 4).

Termination of multi-exit RMDPs, when cast as a model checking problem over GPL along the same lines as our treatment of 1-exit RMDPs, yields an *entangled* GPL formula, which is outside of separable GPL. Thus, separability can be seen as a characteristic of the verification problems that are known to be decidable, when cast in terms of model checking in GPL. Consequently, GPL and its sublogic are useful formalisms to study the relationships between verification problems over branching systems with both probabilistic and internal non-deterministic choice. We discuss these issues in greater detail in Sect. 5.

2 GPL and Branching Systems

In this section, we formally define probabilistic branching systems and give the syntax and semantics of GPL fuzzy formulae.

2.1 Probabilistic Branching Systems

We define a probabilistic branching system (PBS).

▶ **Definition 1** (PBS). With respect to fixed sets Act and $Prop$ of *actions* and *propositions*, respectively, a PBS L is a quadruple (S, δ, P, I), where

- S is a countable set of states;
- $\delta \subseteq S \times Act \times S$ is the transition relation;
- $P : \delta \times \mathbb{N} \to [0, 1]$ is the transition probability distribution satisfying:
  - $\forall s \in S. \forall a \in Act. \forall c \in \mathbb{N}. \sum_{s' : (s,a,s') \in \delta} P(s, a, s', c) \in \{0, 1\}$, and
  - $\forall s \in S. \forall a \in Act. \forall s' \in S. (s, a, s') \in \delta \implies (\exists c \in \mathbb{N}. P(s, a, s', c) > 0)$;
- $I : S \to 2^{Prop}$ is the *interpretation*, recording the set of propositions true at a state.

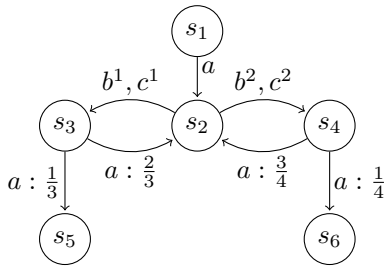

Figure 1 An example PBS.

This definition is in line with that of probabilistic automata [21, 24], in which, given an action, a probabilistic distribution is chosen non-deterministically (we assume there are finitely many distributions for any state-action pair). Other equally expressive models include alternating automata, in which labeled non-deterministic choices are followed by silent probabilistic choices. The difference between such models has been analyzed with respect to bisimulation [25]. However, a PBS is *explicitly* a branching system, as we show next.

Given $L = (S, \delta, P, I)$, a *partial computation* is a sequence $\sigma = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n$, where for all $0 \leq i < n$, $(s_i, a_{i+1}, s_{i+1}) \in \delta$. Also, $\mathsf{fst}(\sigma) = s_0$ and $\mathsf{last}(\sigma) = s_n$ denote, respectively, the first and last states in σ. Each transition of a partial computation is labeled with an action $a_i \in Act$. The set of all partial computations of L is denoted by $\mathcal{C}_L$, and $\mathcal{C}_L(s) = \{\sigma \in \mathcal{C}_L \mid \mathsf{fst}(\sigma) = s\}$. *Composition* of partial computations, $\sigma \xrightarrow{a} \sigma'$, where $\sigma' = s'_0 \xrightarrow{b_1} \cdots \xrightarrow{b_m} s'_m$, represents $s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n \xrightarrow{a} s'_0 \xrightarrow{b_1} \cdots \xrightarrow{b_m} s'_m$ if $(s_n, a, s'_0) \in \delta$. A partial computation σ' is a *prefix* of σ if $\sigma' = s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_i} s_i$ for some $i \leq n$.

From a set of partial computations, we can build deterministic trees. $T \subseteq \mathcal{C}_L$ is *prefix-closed* if, for every $\sigma \in T$ and σ' a prefix of σ, $\sigma' \in T$. T is *deterministic* if for every $\sigma, \sigma' \in T$ with $\sigma = s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n \xrightarrow{a} s \cdots$ and $\sigma' = s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n \xrightarrow{a'} s' \cdots$, either $a \neq a'$ or $s = s'$, i.e., if a pair of computations share a prefix, the first difference cannot involve transitions labeled by the same action. If a tree T has a single starting state, it is denoted $\mathsf{root}(T)$; if $s = \mathsf{root}(T)$ then $T \subseteq \mathcal{C}_L(s)$. We also let $\mathsf{edges}(T) = \{(\sigma, a, \sigma') \mid \sigma, \sigma' \in T \wedge \exists s \in S. \sigma' = \sigma \xrightarrow{a} s\}$.

▶ **Definition 2** (D-trees and outcomes)**.** A d-tree is a set of partial computations with a single starting state that is prefix-closed and deterministic. A d-tree T is maximal if there exists no d-tree T' with $T \subset T'$. An *outcome* is a maximal d-tree.

$\mathcal{T}_L$ refers to all the d-trees of L, and $\mathcal{T}_L(s) = \{T \in \mathcal{T}_L \mid \mathsf{root}(T) = s\}$. T' is a *prefix* of T if $T' \subseteq T$. $T \xrightarrow{a} T'$ means $T' = \{\sigma \mid \mathsf{root}(T) \xrightarrow{a} \sigma \in T\}$. T is *finite* if $|T| < \infty$. $\mathcal{M}_L$ and $\mathcal{M}_L(s)$ are analogous to $\mathcal{T}_L$ and $\mathcal{T}_L(s)$, but for maximal d-trees.

An example PBS and two of its outcomes are shown in Figs. 1 and 2. In the PBS, transitions are usually annotated with their action label and probability (we may omit the probability when it is 1). Note that there are two transitions labeled b (and c) from state s_2, reflecting internal non-determinism, and we use superscripts to distinguish them.

D-trees are constructed from S and δ, without regard for P. A fully probabilistic branching system (fPBS) has no internal non-determinism, i.e., its transition probability distribution P is a function of δ. An fPBS was called an RPLTS in [5], which may have obscured its branching nature.

A property of a PBS will hold for some subset of its outcomes, and we need to measure this set. To resolve the internal non-deterministic choices, we require a scheduler.

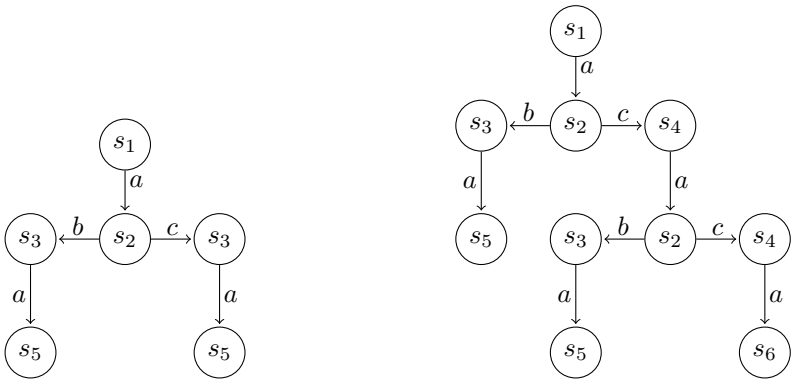

Figure 2 Selected outcomes for the PBS in Fig. 1.

▶ **Definition 3** (Scheduler)**.** A scheduler for a PBS L is a function $\gamma : \mathcal{C}_L \times Act \to \mathbb{N}$, such that if an action a is present at $s = \mathsf{last}(\sigma)$, then $\gamma(\sigma, a) = c$ implies that $\sum_{s'} P(s, a, s', c) = 1$.

Note that we have defined deterministic schedulers, which are also aware of their relevant histories. Given a scheduler γ for a PBS L, we have an fBPS $L_\gamma = (S_\gamma, \delta_\gamma, P_\gamma, I_\gamma)$, where $S_\gamma \subseteq \mathcal{C}_L$ and so $\delta_\gamma \subseteq \mathcal{C}_L \times Act \times \mathcal{C}_L$.

Thus, given a scheduler γ for a PBS L, we can follow the definitions for the measure of d-trees from [5]. A *basic cylindrical subset* of $\mathcal{M}_L(s)$ contains all trees sharing a particular prefix. Letting $s \in S$ and $T \in \mathcal{T}_L(s)$ such that T is finite, $B_T = \{T' \in \mathcal{M}_L \mid T \subseteq T'\}$. Now, we define the measure:

▶ **Definition 4.** For a PBS L with scheduler γ, the probability measure of a basic cylindrical subset B_T is defined by a partial function $\mathsf{m}^\gamma : 2^{\mathcal{M}_L} \to [0, 1]$, where:

$$\mathsf{m}^\gamma(B_T) = \prod_{(\sigma,a,\sigma') \in \mathsf{edges}(T)} P(\mathsf{last}(\sigma), a, \mathsf{last}(\sigma'), \gamma(\sigma, a)) \tag{1}$$

From here, a probability measure $\mathsf{m}^\gamma_s : \mathcal{B}_s \to [0, 1]$ on the smallest σ-field of sets $\mathcal{B}_s$ is generated from basic cylindrical subsets B_T with $\mathsf{m}^\gamma_s(B_T) = \mathsf{m}^\gamma(B_T)$ (cf. [5, Definition 8]).

▶ **Example 5.** Letting $\sigma = s_1 \xrightarrow{a} s_2$, the measure of the left outcome in Figure 2 is $\frac{1}{9}$ for any scheduler γ where $\gamma(\sigma, b) = \gamma(\sigma, c) = 1$, and 0 for all other schedulers.

2.2 GPL Syntax

GPL *fuzzy formulae* depend on *outcomes*. We give the syntax of GPL fuzzy formulae ψ, with $A \in Prop$,[1] $X \in Var$, $a \in Act$, as:

$$\psi ::= A \mid \neg A \mid X \mid \psi \wedge \psi \mid \psi \vee \psi \mid \langle a \rangle \psi \mid [a]\psi \mid \mu X.\psi \mid \nu X.\psi \ . \tag{2}$$

Operators $\mu X.\psi$ and $\nu X.\psi$ are least and greatest fixed point operators for the "equation" $X = \psi$. As in [5], fuzzy formulae are alternation-free, which allows a simpler semantics for fixed points while retaining the expressiveness required for our applications. Additionally, *diamond* implies *box*: $\langle a \rangle \psi$ means that there is an a-transition and it satisfies ψ; $[a]\psi$ means that if there is an a-transition, it satisfies ψ. We may also use a set $\alpha \subseteq Act$ for the modalities, reading $\langle \alpha \rangle \psi$ as $\bigvee_{a \in \alpha} \langle a \rangle \psi$ and $[\alpha]\psi$ as $\bigwedge_{a \in \alpha} [a]\psi$. When we write "$-$" for α, then $\alpha = Act$.

[1] The full syntax allows any state formula [5, 15] as a fuzzy formula.

Table 1 GPL semantics: fuzzy formulae.

$$
\begin{aligned}
\Theta_L(A)e &= \bigcup_{A \in I(s)} \mathcal{M}_L(s), \\
\Theta_L(X)e &= e(X), \\
\Theta_L(\langle a \rangle \psi)e &= \{T \in \mathcal{M}_L \mid \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\psi)e\}, \\
\Theta_L([a]\psi)e &= \{T \in \mathcal{M}_L \mid (T \xrightarrow{a} T') \Rightarrow T' \in \Theta_L(\psi)e\}, \\
\Theta_L(\psi_1 \wedge \psi_2)e &= \Theta_L(\psi_1)e \cap \Theta_L(\psi_2)e, \\
\Theta_L(\psi_1 \vee \psi_2)e &= \Theta_L(\psi_1)e \cup \Theta_L(\psi_2)e, \\
\Theta_L(\mu X.\psi)e &= \bigcup_{i=0}^{\infty} M_i, \text{ where } M_0 = \emptyset \text{ and } M_{i+1} = \Theta_L(\psi)e[X \mapsto M_i], \\
\Theta_L(\nu X.\psi)e &= \bigcap_{i=0}^{\infty} N_i, \text{ where } N_0 = \mathcal{M}_L \text{ and } N_{i+1} = \Theta_L(\psi)e[X \mapsto N_i].
\end{aligned}
$$

2.3 GPL Semantics

We define the semantics of fuzzy formulae with respect to a fixed PBS $L = (S, \delta, P, I)$, where Ψ is the set of all fuzzy formulae. A function $\Theta_L : \Psi \to 2^{\mathcal{M}_L}$, augmented with an extra *environment* parameter $e : Var \to 2^{\mathcal{M}_L}$, returns the set of outcomes satisfying a given fuzzy formula. For a given $s \in S$, $\Theta_{L,s}(\psi) = \Theta_L(\psi) \cap \mathcal{M}_L(s)$. For the fuzzy formulae, the semantics is as in Table 1.

We refer to the value $\sup_\gamma \mathsf{m}^\gamma_s(\Theta_{L,s}(\psi))$ as a *capacity* (following [6]), and write it as $\mathsf{Pr}_{L,s}(\psi)$. In particular, $[a]\psi$ and $\langle a \rangle \psi$ are equivalent when action a is present at a state.

As d-trees do not depend on P, several important properties for GPL over PBSs carry over naturally from [5]. First, we have distributivity on *box* and *diamond* [5, Lemma 1]:

▶ **Lemma 6** (Distributivity on modal operators). *Letting* $\oplus \in \{\wedge, \vee\}$*:*

$$
\begin{aligned}
\Theta_L([a]\psi_1 \oplus [a]\psi_2) &= \Theta_L\big([a](\psi_1 \oplus \psi_2)\big) \\
\Theta_L(\langle a \rangle \psi_1 \oplus \langle a \rangle \psi_2) &= \Theta_L\big(\langle a \rangle(\psi_1 \oplus \psi_2)\big) \\
\Theta_L([a]\psi_1 \wedge \langle a \rangle \psi_2) &= \Theta_L\big(\langle a \rangle(\psi_1 \wedge \psi_2)\big)
\end{aligned} \tag{3}
$$

Similarly, because a PBS L with a scheduler γ yields an fPBS L_γ, we can relate the probability of a conjunction with that of a disjunction and compute the effect of taking a step (where $\gamma^{s'}_a(\sigma, a') = \gamma(s \xrightarrow{a} \sigma, a')$ and $\mathsf{fst}(\sigma) = s'$) [5, Lemma 2]:

$$
\mathsf{m}^\gamma_s(\Theta_{L,s}(\psi_1 \vee \psi_2)) = \mathsf{m}^\gamma_s(\Theta_{L,s}(\psi_1)) + \mathsf{m}^\gamma_s(\Theta_{L,s}(\psi_2)) - \mathsf{m}^\gamma_s(\Theta_{L,s}(\psi_1 \wedge \psi_2)) \tag{4}
$$

$$
\mathsf{m}^\gamma_s(\Theta_{L,s}(\langle a \rangle \psi)) = \sum_{s' : (s,a,s') \in \delta} P(s, a, s', \gamma(s, a)) \cdot \mathsf{m}^{\gamma^{s'}_a}_{s'}(\Theta_{L,s'}(\psi)) \tag{5}
$$

Additionally, we can express the negation of a fuzzy formula ψ, $\mathsf{neg}(\psi)$, such that, for any PBS L and state s ([5, Lemma 3]):

$$
\Theta_{L,s}(\mathsf{neg}(\psi)) = \mathcal{M}_L(s) - \Theta_{L,s}(\psi) \tag{6}
$$

The proof involves switching all the operators to their duals.

▶ **Example 7** (Distributivity and negation). The formula $\psi = \mu X.[a][b]X \wedge [a][c]X$ is satisfied by all *finite* outcomes of the PBS in Fig. 1. By Lemma 6, it is equivalent to $\mu X.[a]([b]X \wedge [c]X)$. Also, $\mathsf{neg}(\psi) = \nu X.\langle a \rangle \langle b \rangle X \vee \langle a \rangle \langle c \rangle X$. Note that a scheduler that maximizes the measure of ψ will minimize the measure of $\mathsf{neg}(\psi)$.

Table 2 Encoding of CTL over PLTSs.

$$E_{CTL}(\psi) = \begin{cases} \psi, & \psi \in \mathit{Prop}, \\ \mathsf{neg}(E_{CTL}(\psi')), & \psi = \neg\psi', \\ E_{CTL}(\psi_1) \wedge E_{CTL}(\psi_2), & \psi = \psi_1 \wedge \psi_2, \\ \langle - \rangle E_{CTL}(\psi), & \gamma = \mathsf{EX}\psi, \\ \mu X. E_{CTL}(\psi_2) \vee (E_{CTL}(\psi_1) \wedge [-]X), & \psi = \mathsf{A}[\psi_1 \mathsf{U} \psi_2], \\ \mu X. E_{CTL}(\psi_2) \vee (E_{CTL}(\psi_1) \wedge \langle - \rangle X), & \psi = \mathsf{E}[\psi_1 \mathsf{U} \psi_2]. \end{cases}$$

3 Encoding Other Model Checking Problems

When GPL was originally introduced, its most interesting known application was to PCTL* model checking [2], as it is straightforward to encode any LTL formula as a fuzzy formula in GPL [5]. In this section, we relate GPL to two applications involving branching systems.

3.1 PTTL and Branching Processes

As GPL fuzzy formulae are interpreted over outcomes of a system, they can encode LTL [5, Sect. 3.2], in the fragment of GPL for systems with $\mathit{Act} = \{a\}$ and no terminal states; then the outcomes are paths, rather than trees. When Act is not limited to one action, so the outcomes are trees, it is correspondingly straightforward to encode CTL (Table 2). This may seem either obvious or strange, as GPL and CTL are both interpreted over trees, but CTL is typically understood as a logic over systems and not their outcomes; this encoding also reduces to the referenced LTL encoding when all the outcomes are paths.

Along similar lines, Probabilistic Tree Temporal Logic (PTTL), has been independently introduced [4]. Branching Processes (BPs) are a branching extension of Markov chains, and PTTL is a logic over BPs. PTTL's fuzzy formulae are limited to the A and E versions of the X, U, and R operators being applied to state formulae, a subset of our CTL encoding.

Additionally, BPs have been extended with non-determinism, yielding Branching MDPs (BMDPs), for which the extinction and reachability problems have been analyzed [10, 12]. When we use only the modal operators $[-]$ and $\langle - \rangle$, as with CTL and PTTL, PBSs are equivalent to BMDPs (and fPBSs to BPs).

3.2 Encoding of RMDP Termination

A Recursive MDP (RMDP) [12] is specified as a set of components. A component has an entry node and one or more exit nodes. Components may contain *boxes*, each box having a *call port* that represents a procedure call and *return ports* to represent a possible return from the called procedure. We provide the formal definition of the (more general) *Recursive Simple Stochastic Games* (RSSGs) [12].

▶ **Definition 8.** An RSSG A is a tuple $(A_1, \ldots, A_k)$, where each *component graph* A_i is a septuple $(N_i, B_i, Y_i, \mathsf{En}_i, \mathsf{Ex}_i, \mathsf{pl}_i, \delta_i)$:

- N_i is a set of nodes, containing subsets En_i and Ex_i of entry and exit nodes, respectively.
- B_i is a set of boxes, with a mapping $Y_i : B_i \to \{1, \ldots, k\}$ assigning each box to a component. Each box has a set of call and return ports, associated with the entry and

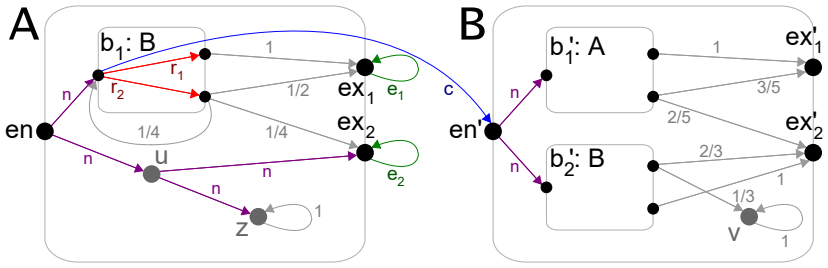

Figure 3 Example RMDP with Call, Return, and Exit edges added to Component "A".

exit nodes, respectively, in the corresponding components: $\mathsf{Call}_b = \{(b, en) \mid en \in \mathsf{En}_{Y_i(b)}\}$, $\mathsf{Return}_b = \{(b, ex) \mid ex \in \mathsf{Ex}_{Y_i(b)}\}$. Additionally, we have:

$$\mathsf{Call}^i = \bigcup_{b \in B_i} \mathsf{Call}_b, \quad \mathsf{Return}^i = \bigcup_{b \in B_i} \mathsf{Return}_b, \quad Q_i = N_i \cup \mathsf{Call}^i \cup \mathsf{Return}^i.$$

- $\mathsf{pl}_i : Q_i \to \{0, 1, 2\}$ is a mapping that specifies whether, at each state, the choice is probabilistic (i.e., *player* 0), or non-deterministic (player 1: maximizing, player 2: minimizing). As any $u \in \mathsf{Call}^i \cup \mathsf{Ex}_i$ has no outgoing transitions, let $\mathsf{pl}_i(u) = 0$ for these states.
- δ_i is the transition relation, with transitions of the form (u, p_{uv}, v), when $\mathsf{pl}_i(u) = 0$ and u is not an exit node or a call port, and v may not be an entry node or a return port. Additionally, $p_{uv} \in (0, 1]$ and, for each u, $\sum_{v':(u,\cdot,v') \in \delta_i} p_{uv'} = 1$. Meanwhile, when $\mathsf{pl}_i(u) > 0$, transitions are of the form $(u, \bot, v)$.

RMDPs only have a player 1 or player 2, depending on whether they are maximizing or minimizing, respectively. Figure 3 shows an RMDP with two components, A and B. Any call to A non-deterministically results in either a call to B (via box b_1) or a transition to u.

3.2.1 Translating RMDPs to PBSs

We can translate an RMDP into a PBS L with $Act = \{p, n, c, r_j, e_j\}$, with states of the PBS corresponding to nodes of the RMDP. We retain the RMDP's transitions, labeling them as "n" for actions from a non-deterministic choice and "p" for probabilistic choice. To this basic structure we add three new kinds of edges:

- "e_j" for the jth exit node of a component,
- "c" edges from a call port to the called component's entry node, and
- "r_j" edges from a call port to each return port in the box.

Figure 3 shows the new edges added to component A (r_1 and r_2 are *distinct* actions here). Formally, we define the translation as follows:

▶ **Definition 9** (Translated RMDP). The translated RMDP A is a PBS $L = (S, \delta, P, I)$:

- The set of states S is the set of all the nodes, as well as the call and return ports of the boxes, i.e., $S = \bigcup_i Q_i$. Additionally, we associate a consistent index with each state corresponding to an exit node or a return port.
- The transition relation δ has all the transitions of the components, labeled by action p for the probabilistic transitions and n for the non-deterministic ones. Thus, when $(u, p_{uv}, v) \in \delta_i$ for any i, then $(u, p, v) \in \delta$, and when $(u, \bot, v) \in \delta_i$, $(u, n, v) \in \delta$. Additionally, we have $((b, en), c, en) \in \delta$ and $((b, en), r_j, (b, ex_j)) \in \delta$ for every box b, and $(ex_j, e_j, ex_j) \in \delta$ for every exit node. Note the indices used, matching r_j with e_j.

- The transition probability distribution P is defined as follows:
  - $P(u, p, v, 0) = p_{uv}$.
  - Assuming a one-to-one function $num : S \to \mathbb{N}$, $P(u, n, v, num(v)) = 1$.
  - When $a \in \{c, r_j, e_j\}$ and $(u, a, v) \in \delta$, $P(u, a, v, 0) = 1$.
- $I(s) = \emptyset$ for all states $s \in S$.

While c edges denote control transfer due to a call, r edges summarize returns from the called procedure (they are similar to the *jump edges* in nested state machines [1]). Additionally, if an RMDP has no non-deterministic choices, it is called a Recursive Markov Chain (RMC), and it can be translated to an fBPS.

3.2.2 GPL Formula for RMDP Termination

We define a formula for an m-exit RMDP, which we will show models termination. In this section, we also write a mu-calculus formula via equations, where, e.g., if $\psi_1 =_\mu f_1(\psi_1, \psi_2)$ and $\psi_2 =_\mu f_2(\psi_1, \psi_2)$, then ψ_1 is $\mu X_1.f_1(X_1, \mu X_2.f_2(X_1, X_2))$.

▶ **Definition 10** (RMDP termination formula)**.** For an m-exit RMDP, the termination formula may be given as m equations, for $1 \leq i \leq m$:

$$\psi_i =_\mu \langle e_i\rangle \mathsf{tt} \vee \langle p\rangle\psi_i \vee \langle n\rangle\psi_i \vee \left(\bigvee_{j=1}^{m} \langle c\rangle\psi_j \wedge \langle r_j\rangle\psi_i\right) \tag{7}$$

We show what the formula looks like for 1 and 2 exits ((8) and (9), respectively):

$$\psi_1 = \mu X.\langle e_1\rangle \mathsf{tt} \vee \langle p\rangle X \vee \langle n\rangle X \vee (\langle c\rangle X \wedge \langle r_1\rangle X) \tag{8}$$

$$\begin{aligned} \psi_1 &=_\mu \langle e_1\rangle \mathsf{tt} \vee \langle p\rangle\psi_1 \vee \langle n\rangle\psi_1 \vee (\langle c\rangle\psi_1 \wedge \langle r_1\rangle\psi_1) \vee (\langle c\rangle\psi_2 \wedge \langle r_2\rangle\psi_1) \\ \psi_2 &=_\mu \langle e_2\rangle \mathsf{tt} \vee \langle p\rangle\psi_2 \vee \langle n\rangle\psi_2 \vee (\langle c\rangle\psi_1 \wedge \langle r_1\rangle\psi_2) \vee (\langle c\rangle\psi_2 \wedge \langle r_2\rangle\psi_2) \end{aligned} \tag{9}$$

The termination problem for 1-exit RMDPs is equivalent to that of BMDP extinction. We can see in (8) that the branching occurs at the call port, with the c and r actions.

Termination of multi-exit RMDPs is undecidable, in general [12].

▶ **Theorem 11** (RMDP translation)**.** *Given an m-exit RMDP A, its translated BPS L, and ψ_i as in (7), there is a one-to-one correspondence between the set of paths $\{P_i\}$ through A terminating at exit i and the set of minimal prefixes $\{T_i\}$ of outcomes of L satisfying ψ_i.*

Proof sketch. First, there is indeed a minimal prefix T_i for any outcome satisfying ψ_i, and we can create it by truncating the tree after any e action and pruning the "incorrect" r_j branches (as $\psi_1, \psi_2, \ldots, \psi_m$ are all mutually exclusive, $\langle c\rangle\psi_j$ is satisfied for exactly one j at each branching node of the outcome). Also, probabilistic and (internal) non-deterministic choices are made identically. We induct on the recursion depth of P_i.

If the depth of P_i is 0, then it never reaches a call port, and T_i is a path taking p and n transitions until exit i, with a concluding e_i transition.

Now, suppose that, for $1 \leq i \leq m$, for all paths P_i with depth $k < d$, we have a matching T_i. We show that for any path P_i with depth d, we also have a matching T_i.

In T_i, a node of the form (b, en) has a c transition to en and an r_j transition to (b, ex_j). The segment of P_i from en to ex_j is a path P'_j. P'_j must have depth $k < d$; so, by the induction assumption, T_i has the matching T'_j as the subtree rooted at en (after the c transition).

Then, P_i continues from (b, ex_j), which T_i matches by taking the r_j transition. Thus, T_i matches recursive calls with c transitions and, for the top level, has a partial computation taking p, n, and r_j transitions until exit i, with a concluding e_i transition. ◀

▶ **Corollary 12** (Undecidability). *GPL model checking over PBSs is undecidable.*

Proof. We can encode an undecidable problem, termination of multi-exit RMDPs, in GPL. It remains to show that the BPS scheduler has sufficient information. This follows because each partial computation in the minimal prefix includes the entire path through the RMDP to reach a particular state, *except* for the calls that already returned, which are represented by r_j transitions instead. So, the scheduler knows the current state and all of the open calls. ◀

4 Decidable Model Checking

The *modchk-fuzzy* procedure from [5, Sect. 4.1] finds the probability of a GPL fuzzy formula over a fixed fPBS. In that case, we can eliminate top-level disjunctions, via (4). In general, though, we would want the relation in (10).

$$\mathsf{Pr}_{L,s}(\psi_1 \vee \psi_2) \stackrel{?}{=} \mathsf{Pr}_{L,s}(\psi_1) + \mathsf{Pr}_{L,s}(\psi_2) - \mathsf{Pr}_{L,s}(\psi_1 \wedge \psi_2) \tag{10}$$

This requires that the same scheduler maximize ψ_1, ψ_2, $\psi_1 \wedge \psi_2$, and $\psi_1 \vee \psi_2$; these may all be distinct. Later, *modchk-fuzzy* was also adapted for systems with internal non-determinism, but limited to non-recursive fuzzy formulae [26]. This is inadequate if we want to model check LTL over MDPs or find the maximum probability of termination in a 1-exit RMDP.

4.1 Graph Construction

Disjunctions in themselves are not the problem, of course. We can still attempt a standard graph construction and see whether we can evaluate the parts of the formula dependent on the current state and cleanly separate the parts dependent on the (branching) future. In this section, we describe this construction.

First, we define a few things needed for the construction. Let $\psi[\psi'/X]$ represent the formula ψ with all free instances of X replaced with ψ'.

▶ **Definition 13** (Fischer-Ladner closure). Given a (closed) formula ψ, its *Fischer-Ladner closure*, $Cl(\psi)$, is the smallest set such that the following hold:

- $\psi \in Cl(\psi)$.
- If $\psi' \in Cl(\psi)$, then:
  - if $\psi' = \psi_1 \wedge \psi_2$ or $\psi_1 \vee \psi_2$, then $\psi_1, \psi_2 \in Cl(\psi)$;
  - if $\psi' = \langle a \rangle \psi''$ or $[a]\psi''$ for some $a \in Act$, then $\psi'' \in Cl(\psi)$;
  - if $\psi' = \sigma X.\psi''$, then $\psi''[\sigma X.\psi''/X] \in Cl(\psi)$, with σ either μ or ν.

It will be useful to view a fuzzy formula as an *and-or tree*.

▶ **Definition 14** (And-or tree). The and-or tree of a fuzzy formula ψ, $AO(\psi)$ is a node labeled by $\oplus$, where $\oplus \in \{\wedge, \vee\}$, with children $AO(\psi_1)$ and $AO(\psi_2)$ when $\psi = \psi_1 \oplus \psi_2$, and a leaf ψ otherwise.

A formula of the form $\langle a \rangle \psi$ or $[a]\psi$ is called *modal*. We say that ψ' is an *unguarded subformula* of ψ if it is a leaf in $AO(\psi)$. Also, $\mathcal{AO}(S)$ represents the set of and-or trees with elements of a set S as leaves. We want to separate a formula by actions, which we attempt by finding its *factored* form.

▶ **Definition 15** (Formula Transformations)**.**

- The *fixed-point expansion* of ψ, denoted by $FPE(\psi)$, is a formula ψ' obtained by expanding any unguarded subformula of the form $\sigma X.\psi_X$ to $\psi_X[\sigma X.\psi_X/X]$ where $\sigma \in \{\mu, \nu\}$.
- We say that a formula is non-probabilistic if it depends only on the current state, i.e., $A \in Prop$ or of the form $\langle a\rangle\phi$ and $[a]\phi$ for $a \in Act$ and $\phi \in \{\mathsf{tt}, \mathsf{ff}\}$. The *partial evaluation* of a fuzzy formula ψ at state s, denoted by $PE(s, \psi)$, is a formula obtained by evaluating unguarded non-probabilistic subformulae and simplifying the result (i.e., $\psi' \wedge \phi$, where ϕ is non-probabilistic, becomes ψ' if ϕ is true and ff otherwise).
- A *grouping* of a formula ψ, denoted by $GRP(\psi)$, groups modalities in a formula using distributivity. Formally, GRP maps ψ to a ψ' that is equivalent to ψ based on the equivalences in Lemma 6, applied left-to-right as much as possible on the top level (i.e., $\langle a\rangle\psi_1 \wedge \langle a\rangle\psi_2$ becomes $\langle a\rangle(\psi_1 \wedge \psi_2)$; we may also rearrange the tree via commutativity and associativity, e.g., from $\langle a\rangle\psi_1 \wedge (\langle a\rangle\psi_2 \wedge \langle b\rangle\psi_3)$ to $(\langle a\rangle\psi_1 \wedge \langle a\rangle\psi_2) \wedge \langle b\rangle\psi_3)$.

▶ **Definition 16** (Factored form and entanglement)**.** A formula ψ is in factored form if $\psi \in \{\mathsf{tt}, \mathsf{ff}\}$, or if ψ satisfies $\psi = GRP(\psi)$ and all leaves of $AO(\psi)$ are modal.
Also, ψ is *entangled* if it is in factored form and any action guards multiple leaves of $AO(\psi)$.

Given a state s, a formula ψ' can be transformed into a semantically equivalent one ψ'' that is in factored form, $\psi'' = FAC(s, \psi') = GRP\left(PE\big(s, FPE(\psi')\big)\right)$.[2]

▶ **Example 17.** For ψ_1 in (9) and s with outgoing c and r_i actions,
$FAC(s, \psi_1) = (\langle c\rangle\psi_1 \wedge \langle r_1\rangle\psi_1) \vee (\langle c\rangle\psi_2 \wedge \langle r_2\rangle\psi_1)$, which is entangled on c, with leaves $\{\langle c\rangle\psi_1, \langle c\rangle\psi_2, \langle r_1\rangle\psi_1, \langle r_2\rangle\psi_1\}$.

▶ **Definition 18** (Dependency graph)**.** The dependency graph for model checking a formula ψ with respect to a state s in PBS L, denoted by $\mathsf{Dg}(s, \psi)$, is a directed graph (N, E), where *node set* $N \subseteq S \times \mathcal{AO}(Cl(\psi))$, and *edge set* $E \subseteq N \times (Act \cup \{\varepsilon, \varepsilon^\wedge, \varepsilon^\vee\}) \times N$; i.e., the edges are labeled from $Act \cup \{\varepsilon, \varepsilon^\wedge, \varepsilon^\vee\}$. If graph construction succeeds, the sets N and E are the smallest such that:

- $(s, \psi) \in N$.
- If $(s', \psi') \in N$ and $\psi' = FAC(s', \psi')$, then ψ' is not entangled (if ψ' is entangled, graph construction terminates with failure).
- If $(s', \psi') \in N$, $\psi'' = FAC(s', \psi')$, and $\psi' \neq \psi''$: $(s', \psi'') \in N$ and $((s', \psi'), \varepsilon, (s', \psi'')) \in E$.
- If $(s', \psi'_1 \oplus \psi'_2) \in N$ (an *and*-node or an *or*-node), then $(s', \psi'_i) \in N$ for $i = 1, 2$. Moreover, $((s', \psi'_1 \oplus \psi'_2), \varepsilon^\oplus, (s', \psi'_i)) \in E$ for $i = 1, 2$, and $\oplus \in \{\wedge, \vee\}$.
- If $(s', \langle a\rangle\psi') \in N$ (action node), then $(s'', \psi') \in N$ for each s'' such that $(s', a, s'') \in \delta$. Moreover, $((s', \langle a\rangle\psi'), a, (s'', \psi')) \in E$.

When we transform ψ' to the factored form ψ'', the semantics does not change, i.e., $\Theta_{L,s'}(\psi') = \Theta_{L,s'}(\psi'')$. For the formulae in factored form, standard GPL semantics applies (Table 1). Recall that when a is present at state s', then $\langle a\rangle\psi'$ and $[a]\psi'$ are equivalent; thus, we can assume all action nodes to be of the form $(s', \langle a\rangle\psi')$. From these semantics, we also get the relationships for the capacities. Here, $\prod$ is the standard product operator, while $\coprod_{i\in I} x_i = 1 - \prod_{i\in I}(1 - x_i)$.

[2] After applying GRP, we may have a modal leaf $\langle a\rangle\psi'_a \notin \mathcal{AO}(Cl(\psi))$. Then, we may view a as a prefix label on the subtree $AO(\psi'_a) \in \mathcal{AO}(Cl(\psi))$.

▶ **Lemma 19** (Capacities). *Fix* $\mathsf{Dg}(s_0, \psi) = (N, E)$. *The capacity* $\mathsf{Pr}_{L,s}(\psi')$ *for a node* (s, ψ') *is as follows:*

- $\mathsf{Pr}_{L,s}(\mathsf{ff}) = 0$ *and* $\mathsf{Pr}_{L,s}(\mathsf{tt}) = 1$.
- *If* (s, ψ') *is an* and*-node, then:* $\mathsf{Pr}_{L,s}(\psi') = \prod_{((s,\psi'),\varepsilon^\wedge,(s,\psi'_i))\in E} \mathsf{Pr}_{L,s}(\psi'_i)$.
- *If* (s, ψ') *is an* or*-node, then:* $\mathsf{Pr}_{L,s}(\psi') = \coprod_{((s,\psi'),\varepsilon^\vee,(s,\psi'_i))\in E} \mathsf{Pr}_{L,s}(\psi'_i)$.
- *If* (s, ψ') *is an action node, i.e.,* $\psi' = \langle a\rangle\psi'_a$, *then:*

$$\mathsf{Pr}_{L,s}(\psi') = \max_{c\in\mathbb{N}} \sum_{((s,\psi'),a,(s',\psi'_a))\in E} P(s, a, s', c) \cdot \mathsf{Pr}_{L,s'}(\psi'_a) \tag{11}$$

- *The remaining nodes* (s, ψ') *have a unique successor* (s, ψ'') *with* $\mathsf{Pr}_{L,s}(\psi') = \mathsf{Pr}_{L,s}(\psi'')$.

Proof. Most of the cases are straightforward and similar to the GPL model checking algorithm [5, Lemma 8] and a result for two-player stochastic parity games [21, Theorem 4.22]. The *and*-node and *or*-node cases have the product and coproduct, respectively, due to independence. We explain the *action node* case in more detail.

The sum over the probabilistic distribution follows from (5); we explain the internal non-deterministic choice. A PBS scheduler makes a choice for an action given the partial computation σ. Here, this choice is made based on a formula, ψ'_a, to be satisfied. When the graph construction succeeds, this is well defined: given L, s, and ψ, the scheduler can deduce ψ'_a from σ, a la traversal of the dependency graph. ◀

We note that, although a particular choice may maximize $\mathsf{Pr}_{L,s}(\psi')$, this does not mean that the corresponding capacity can be reached by a *memoryless*[3] scheduler, as a scheduler that makes this choice *every time* is not necessarily optimal. Indeed, no optimal scheduler may exist, in which case we would only have ϵ-optimal schedulers for any $\epsilon > 0$ [10, 21]. The capacity may be predicated on *eventually* making a different choice. The formulation in Lemma 19 is consistent with this possibility, and the existence of (ϵ-)optimal schedulers may be justified through a method called *strategy improvement* or *strategy stealing* [12, 21]. The intuition is that, in case of a loop, we can add a choice to succeed or fail immediately, succeeding with probability equal to the corresponding capacity. This does not increase the capacity, and the maximizing scheduler can otherwise be the same, if this choice does not arise.

▶ **Theorem 20** (Model checking termination). *The graph construction of* $\mathsf{Dg}(s, \psi)$ *terminates for any fuzzy formula* ψ *and (finite) PBS* L.

Proof. Letting $c = |Cl(\psi)|$, the number of distinct formulae not in factored form is bounded by 2^{2^c} (since we may check DNF versions for equivalence). The number of actions in L and ψ is finite, which bounds the number of action nodes. Thus, construction cannot continue indefinitely. ◀

4.2 Separability of Fuzzy Formulae

Now, we describe a class of fuzzy formulae that cannot get entangled, regardless of the PBS, which we denote as *separability*.

[3] We consider a scheduler memoryless if it makes a choice based on a formula to be satisfied, but does not know the history.

Recall that we used fixed-point expansion, partial evaluation, and grouping to get a factored form. As partial evaluation replaced non-probabilistic formulae with either tt or ff, we define a "worst-case" scenario.

▶ **Definition 21** (Purely probabilistic abstraction)**.** The *purely probabilistic abstraction* of a fuzzy formula ψ, denoted by $PPA(\psi)$, is a formula obtained by removing unguarded non-probabilistic subformulae (i.e., $\psi' \wedge \phi$, where ϕ is non-probabilistic, becomes ψ').

▶ **Definition 22** (Separability)**.** The set of all separable formulae is the largest set $\mathcal{S}$ such that $\forall \psi \in \mathcal{S}$, if $\psi' = GRP(PPA(FPE(\psi)))$, then

1. ψ' is not entangled, and
2. for each leaf $\langle a \rangle \psi_a$ of $AO(\psi')$, $\psi_a \in S$.

A formula ψ is *separable* if $\psi \in \mathcal{S}$.

As separable formulae preclude entanglement, graph construction is guaranteed to succeed, and we have the following result.

▶ **Corollary 23** (Model checking separable formulae)**.** *The graph construction of* $\mathsf{Dg}(s, \psi)$ *terminates* without failure *for any separable fuzzy formula* ψ *and (finite) PBS* L*.*

The decidable problems in Sect. 3 involve separable fuzzy formulae. In the case of LTL, there is only one action, in which case any formula in factored form is a single leaf. Though a CTL formula may be entangled, all PTTL fuzzy formulae are separable, as there is no explicit conjunction or disjunction; thus, we can model check PTTL over BMDPs. Finally, for 1-exit RMDP termination, the same separable formula, (8), is used for any 1-exit RMDP.

4.3 Solving the Polynomial System

For model checking separable formulae, we show how, given the graph, to compare the probabilistic value of ψ at a state s against a threshold p. We do this by first constructing a *system of polynomial max fixed point equations* from the graph. Each node i in the dependency graph is associated with a real-valued variable x_i. Given a set of variables V, each equation in the system is of the form $x_i = e$ where e is

- a polynomial over V such that the sum of coefficients is ≤ 1; or
- of the form $\max(V')$ where $V' \subseteq V$.

Furthermore, the equations form a stratified system, where each variable x_i can be assigned a stratum $j = \mathit{stratum}(x_i)$ such that x_i is defined in terms of only variables of the form x_k such that $\mathit{stratum}(x_k) \leq \mathit{stratum}(x_i)$ (cf. [19, Def. 9]); and variables in the same stratum j fall under the same fixed point. Let $\bowtie \in \{>, \geq, <, \leq\}$.

▶ **Theorem 24.** *Given a real value* p*, a system of polynomial max fixed point equations and a distinguished variable* x *defined in the system, whether or not* $x \bowtie p$ *in its solution is decidable.*

Proof. We write the polynomial system, $\mathbf{x} = P(\mathbf{x})$, as a sentence in the first-order theory of real closed fields, similar to [19]. The additional comparison will be $x_0 \bowtie p$. Along with the equation system, we need to encode fixed points and max.

We can encode $x_i = \max(x_j, x_k)$ as (12) (cf. [12, Section 5]):

$$x_i \geq x_j \wedge x_i \geq x_k \wedge (x_i \leq x_j \vee x_i \leq x_k) \ . \tag{12}$$

Meanwhile, letting V be the set of all variables and I a subset belonging to some stratum with least fixed point, we can encode the fixed point itself as (13):

$$\forall \mathbf{x}'_I. \left(\bigwedge_{i \in I} x'_i = P_i(\mathbf{x}'_I, \mathbf{x}_{V \setminus I}) \implies \bigwedge_{i \in I} x_i \leq x'_i \right) . \tag{13}$$

The stratification of fixed points in the equation system precludes a cyclical dependency between a least and a greatest fixed point; a greatest fixed point can be encoded similarly.

The original fixed point equation system, along with the query $x \bowtie p$, (12)-(13), and the counterpart encoding the greatest fixed point, are sentences in a first order theory of real closed fields, which is decidable [27]. Hence the decidability of $x \bowtie p$ in the solution to the fixed point equations follows. ◀

We use the above result to determine whether or not $\mathsf{Pr}_{L,s}(\psi) \bowtie p$ for a separable formula ψ. The polynomial fixed point system is derived similarly to [5, Section 4.1.2], with a variable $x_{(s,\psi)}$ for each node (s, ψ) in the dependency graph $\mathsf{Dg}(s, \psi)$, and equations based on Lemma 19.

- If ψ is not in factored form, then (s, ψ) has a unique edge labeled by ε to a node (s, ψ'), and $x_{(s,\psi)} = x_{(s,\psi')}$.
- $x_{(s,\mathsf{ff})} = 0$ and $x_{(s,\mathsf{tt})} = 1$.
- If (s, ψ) is an *and*-node, then $x_{(s,\psi)} = \prod_{((s,\psi),\varepsilon^\wedge,(s,\psi_i)) \in E} x_{(s,\psi_i)}$.
- If (s, ψ) is an *or*-node, then $x_{(s,\psi)} = \coprod_{((s,\psi),\varepsilon^\vee,(s,\psi_i)) \in E} x_{(s,\psi_i)}$.
- If (s, ψ) is an action node and $\psi = \langle a \rangle \psi_a$, then

$$x_{(s,\psi)} = \max_{c \in \mathbb{N}} \sum_{((s,\psi),a,(s',\psi_a)) \in E} P(s, a, s', c) \cdot x_{(s',\psi_a)}. \tag{14}$$

▶ **Theorem 25** (Correctness). *The construction of the dependency graph* $\mathsf{Dg}(s, \psi)$*, when* ψ *is separable, yields a polynomial max fixed point equation system, such that the value of* $x_{(s,\psi)}$ *in its solution is* $\mathsf{Pr}_{L,s}(\psi)$*.*

Proof. The correctness result follows from Lemma 19 and the semantics of fixed points given by (13) (and its counterpart). ◀

4.4 Model Checking Example

▶ **Example 26** (Model Checking). For the PBS L in Fig. 1 and fuzzy formula $\psi = \mu X.[a][b]X \wedge [a][c]X$, we have symmetric non-deterministic choices on b and c from state s_2, and the formula is satisfied by all finite d-trees (since both s_3 and s_4 have a probability greater than $\frac{1}{2}$ of returning to s_2, the infinite d-trees have positive measure on any scheduler). Letting $\psi_{bc} = [b]\psi \wedge [c]\psi$, we get the dependency graph shown in Fig. 4.

We find $\mathsf{Pr}_{L,s_1}(\psi)$ as the value of x_1^a in the least fixed point from the following equations:

$$\begin{array}{llll} x_1^a = x_2^{bc} & x_2^b = \max(x_3^a, x_4^a) & x_3^a = \frac{1}{3}x_5^{bc} + \frac{2}{3}x_2^{bc} & x_5^{bc} = 1 \\ x_2^{bc} = x_2^b \cdot x_2^c & x_2^c = \max(x_3^a, x_4^a) & x_4^a = \frac{1}{4}x_6^{bc} + \frac{3}{4}x_2^{bc} & x_6^{bc} = 1 \end{array} \tag{15}$$

Solving the equations, we get $\mathsf{Pr}_{L,s_1}(\psi) = x_1^a = \frac{1}{4}$. (Also, $\mathsf{Pr}_{L,s_1}(\mathsf{neg}(\psi)) = \frac{8}{9}$.)

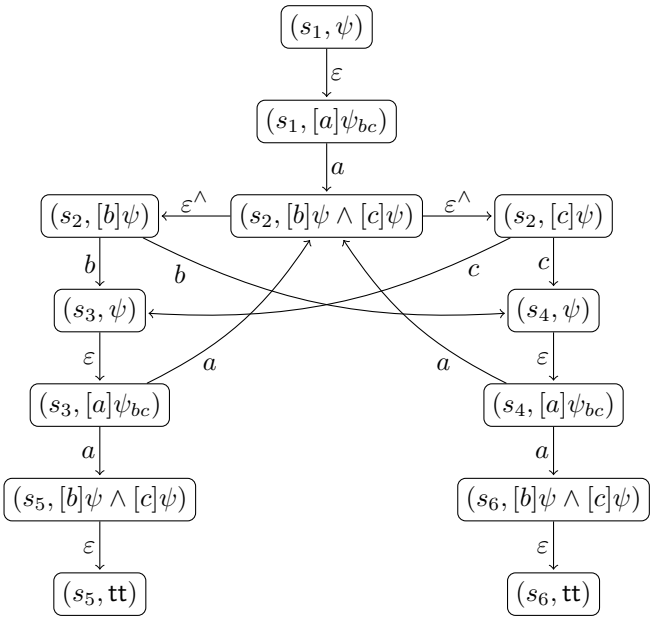

Figure 4 GPL Model Checking Example: Dependency Graph.

5 Discussion and Future Work

Many interesting questions relevant to GPL have been raised in other contexts. Also, while the syntax and semantics of GPL fuzzy formulae remains identical to [5] when over BPSs, the state formulae require a change in syntax; with the resulting logic called XPL in our related extended version [15] ([26] called its extension EGPL).

A probabilistic extension of μ-calculus closely related to separable GPL is Mio's pLμ [20, 21]. In contrast to GPL, the most expressive version of pLμ, denoted pL$\mu^{\odot}_{\oplus}$ [20, 21], defines *three* conjunction operators and their duals such that their probability values can be computed from the probabilities of the conjuncts. The logic pL$\mu^{\odot}$ is able to produce branching systems and supports an intuitive game semantics [21]. Along the same lines as our GPL encoding, we can encode termination of 1-exit RMDPs as model checking in pL$\mu^{\odot}$, and RMC termination in pL$\mu^{\odot}_{\oplus}$. However, pLμ cannot easily encode LTL and has no entanglement, so that attempting to encode multi-exit RMDP termination in pL$\mu^{\odot}_{\oplus}$ similarly to multi-exit RMC termination would lead to an incorrect, rather than undecidable, encoding. Other recent extensions of μ-calculus include the Lukasiewicz μ-calculus [22], μ^p-calculus [3], PμTL [17], and the quantitative μ-calculi, such as qMμ [18] and Qμ [13]. PμTL's expressiveness is orthogonal even with PCTL; the others are more similar to pLμ than GPL.

Although closely related, algorithms to check properties of RMCs (and pPDSs [7]) were developed independently [11]. These were related to algorithms for computing properties of systems such as branching process (BP) extinction and the language probability of Stochastic Context Free Grammars. The relationship between GPL and these systems was mentioned briefly in [16], but has remained largely unexplored. The results of this paper together with [16] also suggest that a separable GPL model checker can be encoded as a probabilistic logic program, even over systems with internal non-determinism.

There has been significant interest in the study of branching systems that also have non-deterministic choices, such as RMDPs and Branching MDPs (BMDPs) [12]. At the same time, the understanding of the polynomial systems has expanded. In [8], the class of

Probabilistic Polynomial Systems (PPS) is introduced, which characterizes when efficient solutions to polynomial equation systems are possible even in the worst case [9]. While [11] did not distinguish the systems arising from 1-exit RMCs from those from multi-exit RMCs, the PPS class is limited to 1-exit RMCs. It was also extended for 1-exit RMDP termination and, later, BMDP reachability, both having polynomial-time complexity for min/maxPPSs [8, 10].

We note that 1-exit RMDP termination, which always has optimal memoryless schedulers, can be matched with the class of least fixed point GPL formulae without disjunctions; meanwhile, BMDP reachability, for which only ϵ-optimal maximizing schedulers may exist, corresponds to least fixed point formulae without conjunctions. Additionally, these systems have been further extended into *stochastic games*, such as Recursive and Branching Simple Stochastic Games (RSSGs and BSSGs, respectively) [12], and we should be able to model these via separable GPL, as well, with a suitably extended BPS. Then, graph construction should be the same, but both min and max operators could appear in the equation system.

Polynomial systems equivalent to those arising from separable GPL over fPBSs have recently been considered in a more general setting in the context of *game automata* [19], followed by an undecidability result for more general properties on the automata [23]. Characterizing equation systems that arise from separable formulae and investigating their efficient solution is an interesting open problem. Another recent result suggests that the alternation-free restriction may be lifted [14]. Finally, this paper addressed the decidability of model checking; determining the complexity of model checking is a topic of future research.

References

1. Rajeev Alur, Swarat Chaudhuri, and P. Madhusudan. Software model checking using languages of nested trees. *ACM Trans. Program. Lang. Syst.*, 33(5):15:1–15:45, November 2011.
2. Christel Baier. On algorithmic verification methods for probabilistic systems. Habilitation thesis, Fakultät für Mathematik & Informatik, Universität Mannheim, 1998.
3. Pablo Castro, Cecilia Kilmurray, and Nir Piterman. Tractable probabilistic mu-calculus that expresses probabilistic temporal logics. In *STACS*, volume 30 of *LIPIcs*, pages 211–223. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.
4. Taolue Chen, Klaus Dräger, and Stefan Kiefer. Model checking stochastic branching processes. In *MFCS*, pages 271–282, Berlin, Heidelberg, 2012. Springer.
5. Rance Cleaveland, S. Purushothaman Iyer, and Murali Narasimha. Probabilistic temporal logics via the modal mu-calculus. *Theoretical Computer Science*, 342(2-3):316–350, 2005.
6. Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Weak bisimulation is sound and complete for PCTL*. In *CONCUR*, volume 2421 of *LNCS*, pages 355–370. Springer Berlin Heidelberg, 2002.
7. Javier Esparza, Antonín Kucera, and Richard Mayr. Model checking probabilistic pushdown automata. In *LICS*, pages 12–21, 2004.
8. Kousha Etessami, Alistair Stewart, and Mihalis Yannakakis. Polynomial time algorithms for branching Markov decision processes and probabilistic min(max) polynomial Bellman equations. In *ICALP, Part I*, pages 314–326, Berlin, Heidelberg, 2012. Springer.
9. Kousha Etessami, Alistair Stewart, and Mihalis Yannakakis. Polynomial time algorithms for multi-type branching processes and stochastic context-free grammars. In *STOC*, pages 579–588. ACM, 2012.
10. Kousha Etessami, Alistair Stewart, and Mihalis Yannakakis. Greatest fixed points of probabilistic min/max polynomial equations, and reachability for branching Markov decision processes. In *ICALP, Part II*, pages 184–196, Berlin, Heidelberg, 2015. Springer.

11 Kousha Etessami and Mihalis Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1):1:1–1:66, February 2009.

12 Kousha Etessami and Mihalis Yannakakis. Recursive Markov decision processes and recursive stochastic games. *J. ACM*, 62(2):11:1–11:69, May 2015.

13 Diana Fischer, Erich Grädel, and Łukasz Kaiser. Model checking games for the quantitative μ-calculus. *Theory of Computing Systems*, 47(3):696–719, 2010.

14 Tomasz Gogacz, Henryk Michalewski, Matteo Mio, and Michał Skrzypczak. Measure properties of regular sets of trees. *Information and Computation*, 256:108 – 130, 2017. URL: `http://www.sciencedirect.com/science/article/pii/S0890540117300627`, `doi:https://doi.org/10.1016/j.ic.2017.04.012`.

15 Andrey Gorlin and C. R. Ramakrishnan. XPL: an extended probabilistic logic for probabilistic transition systems. *CoRR*, abs/1604.06118, 2016.

16 Andrey Gorlin, C. R. Ramakrishnan, and Scott A. Smolka. Model checking with probabilistic tabled logic programming. *TPLP*, 12(4-5):681–700, 2012.

17 Wanwei Liu, Lei Song, Ji Wang, and Lijun Zhang. A simple probabilistic extension of modal mu-calculus. In *IJCAI*, 2015.

18 Annabelle McIver and Carroll Morgan. Results on the quantitative μ-calculus qMμ. *ACM Trans. Comput. Logic*, 8(1), January 2007.

19 Henryk Michalewski and Matteo Mio. On the problem of computing the probability of regular sets of trees. In *FSTTCS*, volume 45 of *LIPIcs*, pages 489–502. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

20 Matteo Mio. Game semantics for probabilistic modal mu-calculi. PhD thesis, The University of Edinburgh, 2012.

21 Matteo Mio. Probabilistic modal μ-calculus with independent product. *Logical Methods in Computer Science*, Volume 8, Issue 4, November 2012. URL: `https://lmcs.episciences.org/789`, `doi:10.2168/LMCS-8(4:18)2012`.

22 Matteo Mio and Alex Simpson. Łukasiewicz mu-calculus. In *FICS*, volume 126 of *EPTCS*, pages 87–104, 2013.

23 Marcin Przybylko and Michal Skrzypczak. On the complexity of branching games with regular conditions. In *LIPIcs*, volume 58. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

24 Roberto Segala. A compositional trace-based semantics for probabilistic automata. In *CONCUR*, pages 234–248, 1995.

25 Roberto Segala and Andrea Turrini. Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In *QEST*, pages 44–53. IEEE Computer Society, 2005.

26 Arvind Soni. Probabilistic and nondeterministic systems. Masters thesis, North Carolina State University, 2004.

27 Alfred Tarski. A decision method for elementary algebra and geometry. *Bulletin of the American Mathematical Society*, 59, 1951.