Abstract

Generalized Probabilistic Logic (GPL) is a temporal logic, based on the modal mu-calculus, for specifying properties of reactive probabilistic systems. We explore XPL, an extension to GPL allowing the semantics of nondeterminism present in Markov decision processes (MDPs). XPL is expressive enough that a number of independently studied problems—such as termination of Recursive MDPs (RMDPs), PCTL* model checking of MDPs, and reachability for Branching MDPs—can all be cast as model checking over XPL. Termination of multi-exit RMDPs is undecidable; thus, model checking in XPL is undecidable in general. We define a subclass, called separable XPL, for which model checking is decidable. Decidable problems such as termination of 1-exit RMDPs, PCTL* model checking of MDPs, and reachability for Branching MDPs can be reduced to model checking separable XPL. Thus, XPL forms a uniform framework for studying problems involving systems with non-deterministic and probabilistic behaviors, while separable XPL provides a way to solve decidable fragments of these problems.

1 Introduction

For finite-state systems, model checking a temporal property can be cast in terms of model checking in the modal \( \mu \)-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. Two of the notable logics for probabilistic systems based on the \( \mu \)-calculus are GPL [1] and pL\( \mu \) [2].

\*This work was partially supported by NSF grant IIS-1447549.
GPL is defined over Reactive Probabilistic Labeled Transition Systems (RPLTSs). In an RPLTS, each state has a set of outgoing transitions with distinct labels; each transition, in turn, specifies a (probabilistic) distribution of target states. The branching-time probabilistic logic GPL is expressive enough to serve as an “assembly language” of a large number of probabilistic temporal logics. For instance, model checking PCTL* properties over Markov Chains, as well as termination and reachability of Recursive Markov Chains can be cast in terms of GPL model checking \[1, 3\].

In this paper, we propose an extension to GPL, which we call Extended Probabilistic Logic (XPL), to express properties of probabilistic systems with internal nondeterministic choice. Where GPL has probabilistic quantifiers \(\Pr_{\geq p} \psi\) over fuzzy formulae \(\psi\), XPL introduces quantifiers over infimum and supremum probabilities, of the form \(\Pr_{\geq p}^\text{min} \psi\). This allows XPL formulae to be quantified over schedulers that resolve internal non-deterministic choices. The semantics of XPL is defined over Probabilistic Labeled Transition Systems (PLTSs). In a PLTS, each state has a set of outgoing transitions, possibly with common labels; and each transition specifies a distribution of target states. PLTSs thus exhibit both probabilistic choice and internal nondeterministic choice.

**Contributions and Significance:** XPL is expressive enough that a wide variety of independently-studied verification problems can be cast as model checking PLTSs with XPL. 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 PLTSs with XPL. We introduce a syntactically-defined subclass, called separable XPL, for which model checking is decidable. We describe a procedure for model checking XPL which always terminates—successfully with the model checking result, or with failure—such that it always terminates successfully for separable XPL (see Sect. 4).

A number of distinct model checking algorithms have been developed independently for decidable verification problems involving systems that have probabilistic and internal non-deterministic choice. Examples of such problems include PCTL* model checking of MDPs [4] and termination of 1-exit RMDPs [6]. These problems can all be reduced, in linear time, to model checking separable XPL formulae over PLTSs (see Sect. 5). To the best of our knowledge, the similarities between these decidable problems and the commonalities between the underlying decision algorithms have not been recognized in the literature.

Termination of multi-exit RMDPs, cast as a model checking problem over XPL along the same lines as our treatment of 1-exit RMDPs, yields an XPL formula that is not separable. 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 XPL. Consequently, XPL in general, and separable XPL in particular, form a useful formalism to study the relationships between verification problems over systems involving probabilistic and internal non-deterministic choice. We discuss these issues in greater detail in Sect. 6.
2 Preliminaries

In this section, we formally define PLTSs, which are used to define the semantics of XPL. We also summarize the syntax and semantics of GPL, using the notations from [1].

2.1 Probabilistic Labeled Transition Systems

We define a probabilistic labeled transition system (PLTS) as an extension of [1]'s RPLTS.

Definition 1 (PLTS). With respect to fixed sets Act and Prop of actions and propositions, respectively, a PLTS $L$ is a quadruple $(S, \delta, P, I)$, where

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

A reactive PLTS does not have internal nondeterminism, i.e., its transition probability distribution $P$ is a function of $\delta$. This definition is in line with the most general for a PLTS [2, 7], in which, given an action, a probabilistic distribution is chosen nondeterministically (we assume that there are finitely many nondeterministic choices). Other equally expressive models include alternating automata, in which labeled nondeterministic ones are followed by silent probabilistic choices. The difference between such models has been analyzed with respect to bisimulation [8].

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, $\text{fst}(\sigma) = s_0$ and $\text{last}(\sigma) = s_n$ denote, respectively, the first and last states in $\sigma$. Each transition of a partial computation is labeled with an action $a_i \in \text{Act}$. The set of all partial computations of $L$ is denoted by $C_L$, and $C_L(s) = \{ \sigma \in C_L \mid \text{fst}(\sigma) = s \}$.

Composition of partial computations, $\sigma \xrightarrow{a} \sigma'$, 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 $\sigma'$ is a prefix of $\sigma$ 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 (d-trees). We often denote a d-tree by the set of paths in the tree. Every d-tree
is prefix-closed and deterministic. $T \subseteq C_L$ is \textit{prefix-closed} if, for every $\sigma \in T$ and $\sigma'$ a prefix of $\sigma$, $\sigma' \in T$. $T$ is \textit{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. A d-tree $T$ has a starting state, denoted root($T$); if $s = \text{root}(T)$ then $T \subseteq C_L(s)$. We also let edges($T$) = \{($\sigma, a, \sigma'$) | $\sigma, \sigma' \in T \land \exists s \in S. \sigma' = \sigma \xrightarrow{a} s$\}.

$T_L$ refers to all the d-trees of $L$, and $T_L(s) = \{T \in T_L \mid \text{root}(T) = s\}$. $T'$ is a prefix of $T$ if $T' \subseteq T$. $T \xrightarrow{a} T'$ means $T' = \{\sigma \mid \text{root}(T) \xrightarrow{a} \sigma \in T\}$. $T$ is \textit{finite} if $|T| < \infty$, and \textit{maximal} if there exists no d-tree $T'$ with $T \subset T'$. $M_L$ and $M_L(s)$ are analogous to $T_L$ and $T_L(s)$, but for maximal d-trees. An \textit{outcome} is a maximal d-tree.
An example PTLS and two of its outcomes are shown in Fig. 1. In the figure, transitions are usually annotated with their action label and probability; the probability is omitted when it is 1. Note that there are two transitions labeled $b$ from state $s_2$ reflecting internal nondeterminism. If we label transition from $s_2$ to $s_3$ only with $b$ (omitting $c$) and that from $s_2$ to $s_4$ only with $c$ (omitting $b$), we get an RPLTS with only probabilistic and external choices.

Note that, with d-trees, we have the distinction between linear- and branching-time semantics for the nondeterministic choices which are internal and external, respectively. Since a d-tree is defined to be deterministic, all of the internal choices (both probabilistic and nondeterministic) are resolved, but the external choices remain. Meanwhile, a property of a PLTS will hold for some subset of its maximal d-trees. In order to give the property a probability, we need a measure of this set. This is straightforward for an RPLTS, as all internal choices are probabilistic; but we will need to do more for PLTSs with internal nondeterministic choice.

Thus, the subsequent concepts apply only to RPLTSs, and we will extend them to PLTSs in Sect. 3. A finite RPLTS d-tree has finite measure, which can be computed from the values of the probabilistic choices in the trees, i.e., its edges. An infinite d-tree will typically have zero measure, but an infinite set of these may have positive measure. Instead, intuitively, we consider the probability of some finite prefix, which again is the product of the probabilities of all the edges. Formally, a basic cylindrical subset of $M_L(s)$ contains all trees sharing a given prefix. Letting $s \in S$, and $T \in T_L(s)$ to be finite, $B_T = \{T' \in M_L \mid T \subseteq T'\}$. The measure of $B_T$ is:

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

From here, a probability measure $m_s : \mathcal{B}_s \to [0,1]$ on the smallest field of sets $\mathcal{B}_s$ is generated from subsets $B_T$ with $m_s(B_T) = m(B_T)$ [Definition 8].

2.2 GPL Syntax

GPL has two different kinds of formulae. State formulae depend directly only on the given state. Fuzzy formulae depend on outcomes. We give the syntax of GPL, with $X \in \text{Var}$, $a \in \text{Act}$, $A \in \text{Prop}$, and $0 \leq p \leq 1$, for state formulae, $\phi$, and fuzzy formulae, $\psi$, as:

$$\phi ::= A \mid \neg A \mid \phi \land \phi \mid \phi \lor \phi \mid \Pr_{>p}\psi \mid \Pr_{\geq p}\psi$$

$$\psi ::= \phi \mid X \mid \psi \land \psi \mid \psi \lor \psi \mid \langle a \rangle \psi \mid [a] \psi \mid \mu X.\psi \mid \nu X.\psi$$

Note that only atomic propositions may be negated, but every operator has its dual given in the syntax. The propositional connectives, $\land$ and $\lor$, can be used on both state and fuzzy formulae. Operators $\mu X.\psi$ and $\nu X.\psi$ are least and greatest fixed point operators for the “equation” $X = \psi$. Additionally, fuzzy formulae must be alternation-free, which prohibits a kind of mixing of least and greatest fixed points, and a formula $\psi$ used to construct state formulae $\Pr_{>p}\psi$ and $\Pr_{\geq p}\psi$. 

\[5\]
in terms of RPLTS d-trees. In that interpretation, diamond

\[ \varphi \]

We define the semantics of GPL with respect to a fixed RPLTS

\[ \Theta_L(\phi)e = \bigcup_{s \models_L \phi} M_L(s), \text{ where } \phi \text{ is a closed formula,} \]

\[ \Theta_L(X)e = e(X), \]

\[ \Theta_L([a] \psi)e = \{ T \in M_L \mid \exists ! T' : T \xrightarrow{a} T' \land T' \in \Theta_L(\psi)e \}, \]

\[ \Theta_L(\exists a \phi)e = \{ T \in M_L \mid (T \xrightarrow{a} T') \Rightarrow T' \in \Theta_L(\phi)e \}, \]

\[ \Theta_L(\psi \land \psi_2)e = \Theta_L(\psi_1)e \cap \Theta_L(\psi_2)e, \]

\[ \Theta_L(\psi \lor \psi_2)e = \Theta_L(\psi_1)e \cup \Theta_L(\psi_2)e, \]

\[ \Theta_L(\mu X \phi)e = \bigcup_{i=0}^{\infty} M_i, \text{ where } M_0 = \emptyset \text{ and } M_{i+1} = \Theta_L(\phi)e[X \mapsto M_i], \]

\[ \Theta_L(\nu X \phi)e = \bigcap_{i=0}^{\infty} N_i, \text{ where } N_0 = M_L \text{ and } N_{i+1} = \Theta_L(\phi)e[X \mapsto N_i]. \]

Table 1: GPL/XPL semantics: fuzzy formulae

| Expression         | Definition                                                                 |
|--------------------|---------------------------------------------------------------------------|
| \( \Theta_L(\phi)e \) | \( \bigcup_{s \models_L \phi} M_L(s) \), where \( \phi \) is a closed formula, |
| \( \Theta_L(X)e \)   | \( e(X) \),                                                                |
| \( \Theta_L([a] \psi)e \) | \( \{ T \in M_L \mid \exists ! T' : T \xrightarrow{a} T' \land T' \in \Theta_L(\psi)e \} \), |
| \( \Theta_L(\exists a \phi)e \) | \( \{ T \in M_L \mid (T \xrightarrow{a} T') \Rightarrow T' \in \Theta_L(\phi)e \} \), |
| \( \Theta_L(\psi \land \psi_2)e \) | \( \Theta_L(\psi_1)e \cap \Theta_L(\psi_2)e \), |
| \( \Theta_L(\psi \lor \psi_2)e \) | \( \Theta_L(\psi_1)e \cup \Theta_L(\psi_2)e \), |
| \( \Theta_L(\mu X \phi)e \) | \( \bigcup_{i=0}^{\infty} M_i, \text{ where } M_0 = \emptyset \text{ and } M_{i+1} = \Theta_L(\phi)e[X \mapsto M_i] \), |
| \( \Theta_L(\nu X \phi)e \) | \( \bigcap_{i=0}^{\infty} N_i, \text{ where } N_0 = M_L \text{ and } N_{i+1} = \Theta_L(\phi)e[X \mapsto N_i] \). |

Table 2: GPL semantics: state formulae

| Expression         | Definition                                                                 |
|--------------------|---------------------------------------------------------------------------|
| \( s \models_L A \) | if \( A \in I(s) \),                                                      |
| \( s \models_L \neg A \) | if \( A \notin I(s) \),                                                  |
| \( s \models_L \phi_1 \land \phi_2 \) | if \( s \models_L \phi_1 \) and \( s \models_L \phi_2 \), |
| \( s \models_L \phi_1 \lor \phi_2 \) | if \( s \models_L \phi_1 \) or \( s \models_L \phi_2 \), |
| \( s \models_L \Pr_{>p} \psi \) | if \( m_\alpha(\Theta_L(s)(\psi)) > p \),                                 |
| \( s \models_L \Pr_{\geq p} \psi \) | if \( m_\alpha(\Theta_L(s)(\psi)) \geq p \).                            |

may not have any free variables. These operators check the probability for a fuzzy formula \( \psi \) (\( \Pr_{>p} \) and \( \Pr_{\geq p} \) are duals). The semantics of GPL is given in terms of RPLTS d-trees. In that interpretation, diamond implies box: \( \langle a \rangle \psi \) means that there is an \( a \)-transition and it satisfies \( \psi \); \( [a] \psi \) means that if there is an \( a \)-transition, it satisfies \( \psi \). We also use a set \( \alpha \subseteq \text{Act} \) for the modalities, reading \( \langle a \rangle \psi \) as \( \bigvee_{\alpha \subseteq \text{Act}} \langle a \rangle \psi \) and \( [a] \psi \) as \( \bigwedge_{\alpha \subseteq \text{Act}} [a] \psi \). When we write “\( \neg \)” for \( \alpha \), that represents \( \text{Act} \).

2.3 GPL Semantics

We define the semantics of GPL with respect to a fixed RPLTS \( L = (S, \delta, P, I) \), where \( \Phi \) and \( \Psi \) are the sets of all state and fuzzy formulae, respectively. A function \( \Theta_L : \Psi \rightarrow 2^{M_L} \), augmented with an extra environment parameter \( e : \text{Var} \rightarrow 2^{M_L} \), returns the set of outcomes satisfying a given fuzzy formula, defined inductively in Table 1.

For a given \( s \in S \), \( \Theta_{L,s}(\psi) = \Theta_L(\psi) \cap M_L(s) \). The relation \( \models_L \subseteq S \times \Phi \) indicates when a state satisfies a state formula, and it is defined inductively in Table 2. Note that the definitions for \( \Theta_L \) and \( \models_L \) are mutually recursive.

There are two properties of GPL fuzzy formulae that are important for the completeness of the GPL model checking algorithm. First, we have distributivity on box and diamond [1] Lemma 1:
Lemma 2 (Distributivity on modal operators). Letting $\oplus \in \{\land, \lor\}$:

\[
\Theta_L([a]\psi_1 \oplus [a]\psi_2) = \Theta_L([a](\psi_1 \oplus \psi_2))
\]
\[
\Theta_L(\langle a \rangle \psi_1 \oplus \langle a \rangle \psi_2) = \Theta_L(\langle a \rangle(\psi_1 \oplus \psi_2))
\]
\[
\Theta_L([a] \psi_1 \land \langle a \rangle \psi_2) = \Theta_L([a](\psi_1 \land \psi_2))
\] (2)

Second, we can relate the probability of a conjunction with that of a disjunction and compute the effect of taking a step [1, Lemma 2]:

\[
m_s(\Theta_{L,s}(\psi_1 \lor \psi_2)) = m_s(\Theta_{L,s}(\psi_1)) + m_s(\Theta_{L,s}(\psi_2)) - m_s(\Theta_{L,s}(\psi_1 \land \psi_2)) \quad (3)
\]
\[
m_s(\Theta_{L,s}(\langle a \rangle \psi)) = \sum_{s' : (s,a,s') \in \delta} P(s,a,s') \cdot m_{s'}(\Theta_{L,s'}(\psi)) \quad (4)
\]

Additionally, although there is no negation operator in the syntax, we can write the negation of a fuzzy formula $\psi$, $\text{neg}(\psi)$, and of a state formula $\phi$, $\text{neg}(\phi)$, such that, for any RPLTS $L$ and state $s$ [1, Lemma 3]):

\[
\Theta_{L,s}(\text{neg}(\psi)) = M_L(s) - \Theta_{L,s}(\psi) \quad \text{and} \quad \models_L \text{neg}(\phi) \iff s \not\models_L \phi.
\]

The proof involves switching all the operators to their duals.

3 XPL

To resolve the nondeterministic transitions in a PLTS, we additionally require a scheduler. Recall, from Sect. 2.1, that $C_L$ is the set of all partial computations $\sigma$ of $L$.

Definition 3 (Scheduler). A scheduler for a PLTS $L$ is a function $\gamma : C_L \times \text{Act} \to \mathbb{N}$.

Note that we have defined deterministic schedulers, which are also aware of their relevant histories. Given a scheduler $\gamma$ for a PLTS $L$, we have a (countable) RPLTS $L_\gamma$, where $S_{L,\gamma} \subseteq C_L$ and so $\delta_{L,\gamma} \subseteq C_L \times \text{Act} \times C_L$. We define a probability distribution:

Definition 4 (Combined probability). The probability distribution of a PLTS $L$ with scheduler $\gamma$ is a function, $P_{L,\gamma} : \delta_{L,\gamma} \to [0,1]$, where:

\[
P_{L,\gamma}(\sigma,a,\sigma') = P_L(\text{last}(\sigma),a,\text{last}(\sigma'),\gamma(\sigma,a)) \quad (5)
\]

We also let $P_{L,\gamma}(\sigma,a,\sigma') = 0$ when $(\sigma,a,\sigma') \notin \delta_{L,\gamma}$.

Recall, from Sect. 2.1, that the basic cylindrical subset $B_T$ contains all maximal d-trees sharing the prefix tree $T$. For these subsets, we define the probability measure:

7
Table 3: XPL semantics: state formulae

| $s \models_{L} A$ | iff $A \in I(s)$, |
| $s \models_{L} \neg A$ | iff $A \notin I(s)$, |
| $s \models_{L} \phi_1 \land \phi_2$ | iff $s \models_{L} \phi_1$ and $s \models_{L} \phi_2$, |
| $s \models_{L} \phi_1 \lor \phi_2$ | iff $s \models_{L} \phi_1$ or $s \models_{L} \phi_2$, |
| $s \models_{L} \Pr_{\text{max}} \psi$ | iff $\sup_{(s, \gamma, s') \in \text{edges}(T)} P_{\gamma}(\sigma, a, \sigma')$ |
| $s \models_{L} \Pr_{\text{min}} \psi$ | iff $\inf_{(s, \gamma, s') \in \text{edges}(T)} P_{\gamma}(\sigma, a, \sigma')$. |

Definition 5 (Probability measure). For a PLTS $L$ with scheduler $\gamma$, the probability measure of a basic cylindrical subset $B_T$ is defined by a partial function $m^\gamma : 2^{M_L} \rightarrow [0, 1]$, where:

$$m^\gamma(B_T) = \prod_{(\sigma, a, \sigma') \in \text{edges}(T)} P_{\gamma, L}(\sigma, a, \sigma')$$  (6)

Since $m^\gamma$ may be considered as defined for an RPLTS, we can extend it to a measure $m^\gamma_s$ as in Sect. 2.1.

3.1 XPL Syntax

Now we give the XPL syntax, with $\triangleright \in \{\triangleright, \triangleright=\}$:

$$\phi ::= A \mid \neg A \mid \phi \land \phi \mid \phi \lor \phi \mid \Pr_{\text{min}} \psi \mid \Pr_{\text{max}} \psi$$

$$\psi ::= \phi \mid X \mid \psi \land \psi \mid \psi \lor \psi \mid \langle a \rangle \psi \mid [a] \psi \mid \mu X. \psi \mid \nu X. \psi$$

The fuzzy formulae remain the same as in GPL. $\Pr_{\text{min}}$ and $\Pr_{\text{max}}$ compare against the infimum and supremum probabilities, respectively, over all schedulers. Note that $\Pr_{\text{min}}$ is the dual of $\Pr_{\text{max}}$. For an RPLTS, $\Pr_{\text{min}}$ and $\Pr_{\text{max}}$ are equivalent.

3.2 XPL Semantics

The semantics of XPL changes from GPL only due to the measure of the PLTS outcomes. In particular, we retain the same semantics on diamond and box. The semantics is defined with respect to a fixed PLTS $L = (S, \delta, P, I)$. The function $\Theta_L : \Psi \rightarrow 2^{M_L}$ remains the same, while $\models_{L} s : S \times \Phi$ differs for the probabilistic operators.

Definition 6 (XPL semantics). The semantics for the state formulae is given in Table 3. For the fuzzy formulae, the semantics are as in Table 1.

Note the use of sup and inf in Table 3. We refer to the value $\sup_{(s, \gamma, s') \in \text{edges}(T)} m^\gamma_s(\Theta_L(s))$ as a probabilistic value and write it as $\Pr_{\gamma, L, s}(\psi)$ (9) calls this a capacity. Unlike in GPL, we may not always be able to compute it with a model checking algorithm.
3.3 Separability of Fuzzy Formulae

With internal nondeterminism, we lose the general relation between conjunctions and disjunctions, as in (3). However, since we are maximizing (or minimizing) over schedulers, we would want the relation in (7).

\[
\Pr_{L,s}^{\text{max}}(\psi_1 \lor \psi_2) \geq \Pr_{L,s}^{\text{max}}(\psi_1) + \Pr_{L,s}^{\text{max}}(\psi_2) - \Pr_{L,s}^{\text{max}}(\psi_1 \land \psi_2)
\] (7)

This requires that the optimal strategy be the same for \(\psi_1, \psi_2, \psi_1 \land \psi_2,\) and \(\psi_1 \lor \psi_2;\) in general, these may all be distinct. Instead, we will seek to delay the application of all conjunctions and disjunctions until the two sides are independent, primarily through repeated application of Lemma 2, which holds for XPL as well because it deals with sets of d-trees, but not their measure. For example, we can rewrite \(\psi_a = \nu X.\langle a \rangle \langle b \rangle X \lor \langle a \rangle \langle c \rangle X\) as \(\nu X.\langle a \rangle(\langle b \rangle X \lor \langle c \rangle X).\) We generalize this to a syntactic notion of separability, defined below. It will be useful to view a fuzzy formula as a kind of an and-or tree.

Definition 7 (And-or tree). The and-or tree of a fuzzy formula \(\psi, AO(\psi)\) is a node labeled by \(\oplus, \) where \(\oplus \in \{\land, \lor\}\) with children \(AO(\psi_1)\) and \(AO(\psi_2)\) when \(\psi = \psi_1 \oplus \psi_2,\) and a leaf \(\psi\) otherwise.

We can flatten this tree with the straightforward flattening operator, where, e.g., the tree \(\land(\psi_1, \ldots, \land(\psi_2, \psi_3))\) may be flattened to \(\land(\psi_1, \ldots, \psi_2, \psi_3).\) Note that flattened trees have alternating \(\land\) and \(\lor\) nodes. A (conjunctive) set of formulae \(F\) corresponds to a flattened and-or tree with the root node labeled by \(\land\) and having the elements of \(F\) as leaves. We will assume \(AO(\psi)\) refers to the flattened tree.

A subformula of \(\psi\) of the form \(\langle a \rangle \psi'\) or \([a] \psi'\) is called a modal subformula of \(\psi.\) We say that \(\psi'\) is an unguarded subformula of \(\psi\) if it is a leaf in \(AO(\psi).\) The GPL model checking algorithm requires bound variables to be guarded by actions (i.e., \(\mu X.(\langle a \rangle X \land \ldots)\) is fine, but \(\mu X.(X \land \ldots)\) is not) \(\llbracket,\) and we adopt this requirement as well.

Definition 8 (Formula Transformations).

- The fixed-point expansion of \(\psi,\) denoted by \(FPE(\psi),\) is a formula \(\psi'\) 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 is a state formula, or of the form \(\langle a \rangle \phi\) and \([a] \phi\) for \(a \in \text{Act}\) and \(\phi \in \{\text{tt, ff}\}.\) The purely probabilistic abstraction of a fuzzy formula \(\psi,\) denoted by \(PPA(\psi),\) is a formula obtained by removing unguarded non-probabilistic subformulae (i.e., \(\psi' \land \phi,\) where \(\phi\) is non-probabilistic, becomes \(\psi'\), etc.).

- A grouping of a formula \(\psi,\) denoted by \(GRP(\psi),\) groups modalities in a formula using distributivity. Formally, \(GRP\) maps \(\psi\) to a \(\psi'\) that is equivalent to \(\psi\) based on the equivalences in Lemma 2 applied left-to-right as much as possible on the top level.
At a high level, a necessary condition of separability is that the actions guarding distinct conjuncts and disjuncts of a formula are distinct as well.

**Definition 9** (Action set). The *action set* of a formula $\psi$, denoted by $\text{action}(\psi)$, is the set of actions appearing at unguarded modal subformulae of $\psi$:

- $\text{action}(\phi) = \emptyset$;
- $\text{action}(\langle a \rangle \psi) = \text{action}([a] \psi) = \{a\}$;
- $\text{action}(\psi_1 \land \psi_2) = \text{action}(\psi_1 \lor \psi_2) = \text{action}(\psi_1) \cup \text{action}(\psi_2)$;
- $\text{action}(\mu X. \psi) = \text{action}(\nu X. \psi) = \text{action}(\psi)$.

We can now define separability based on action sets of formulae as follows.

**Definition 10** (Separability). The set of all separable formulae is the largest set $\mathcal{S}$ such that for all $\psi \in \mathcal{S}$:

1. every subformula of $\psi'$ is in $\mathcal{S}$, and
2. if $\psi' = \psi_1 \oplus \psi_2$ where $\oplus \in \{\land, \lor\}$, then $\text{action}(\psi_1) \cap \text{action}(\psi_2) = \emptyset$.

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

Below we illustrate separability of formulae. Let $\psi_1$-$\psi_4$ be all separable and distinct, and also let $\psi_1 \lor \psi_2$ and $\psi_3 \lor \psi_4$ be separable.

Note that GRP uses only distributivity of the modal operators over "\&" and "\lor", and not the distributivity of the boolean operators themselves. Consequently, a separable formula may be equivalent to a non-separable formula.

**Example 11** (Separable formula with equivalent non-separable formula). The formula $\psi_s$ is separable.

$$\psi_s = [a](\psi_1 \lor \psi_2) \land [b](\psi_3 \lor \psi_4) \quad (8)$$

The DNF version of $\psi_s$, $\psi'_s$, is not separable since action sets of disjuncts overlap.

$$\psi'_s = ([a]\psi_1 \land [b]\psi_3) \lor ([a]\psi_1 \land [b]\psi_4) \lor ([a]\psi_2 \land [b]\psi_3) \lor ([a]\psi_2 \land [b]\psi_4) \quad (9)$$

This is important because we need the subformulae of a separable formula to also be separable.

**Example 12** (Non-separable formula). The formula $\psi_e$ is a subformula of $\psi'_s$, is not separable, and has no equivalent separable formula:

$$\psi_e = ([a]\psi_1 \land [b]\psi_3) \lor ([a]\psi_2 \land [b]\psi_3) \quad (10)$$

With $\psi_e$, we need to satisfy $\psi_1$ or $\psi_2$ following an $a$ action, and likewise for $\psi_3$ or $\psi_4$ following a $b$ action. An equivalent separable formula would thus have to include $[a](\psi_1 \lor \psi_2)$ and $[b](\psi_3 \lor \psi_4)$, but this would also be satisfied by, e.g., outcomes satisfying only $[a]\psi_1 \land [b]\psi_3$. 

10
We say that a formula is entangled at a state if it is not (equivalent to) a separable formula even after considering that state’s specific characteristics. For instance, \( \psi_e \) is entangled only at states with both \( a \) and \( b \) actions present. Even when considering only states where the actions relevant to entanglement are present, a formula may be entangled at some states and not at others.

**Example 13** (Entanglement on \( a \) and \( b \) depends on \( c \)). The formula \( \psi_c \) reduces to \( \psi'_e \) at states that have a \( c \)-transition, and to \( \psi_e \) otherwise.

\[
\psi_c = ([a]\psi_1 \land [b]\psi_3 \land \langle c \rangle tt) \lor ([a]\psi_1 \land [b]\psi_4) \lor ([a]\psi_2 \land [b]\psi_3) \lor ([a]\psi_2 \land [b]\psi_4 \land \langle c \rangle tt). \tag{11}
\]

There are also non-separable formulae that nonetheless would not be entangled at any state of an arbitrary PLTS.

**Example 14** (Never-entangled non-separable formula). For the formula \( \psi_d \), \( \text{PPA}(\psi_d) = \psi_e \), but at any state it is equivalent either to \([a]\psi_1 \land [b]\psi_4 \) or to \([a]\psi_2 \land [b]\psi_3 \).

\[
\psi_d = ([a]\psi_1 \land [b]\psi_4 \land [c]ff) \lor ([a]\psi_2 \land [b]\psi_3 \land \langle c \rangle tt). \tag{12}
\]

Since GRP combines modal subformulae with a common action, we have the following important consequence.

**Remark.** All conjunctive formulae and disjunctive formulae are separable.

## 4 Model Checking XPL Formulae

We outline a model checking procedure for XPL formulae for a fixed PLTS \( L = (S, \delta, P, I) \), along similar lines to the GPL model checking algorithm in [1, Sect. 4]. The model checking procedure succeeds whenever the given formula is separable.

**Definition 15** (Fisher-Ladner closure). Given a formula \( \psi \), its Fisher-Ladner closure, \( \text{Cl}(\psi) \), is the smallest set such that the following hold:

- \( \psi \in \text{Cl}(\psi) \).
- If \( \psi' \in \text{Cl}(\psi) \), then:
  - if \( \psi' = \psi_1 \lor \psi_2 \) or \( \psi_1 \land \psi_2 \), then \( \psi_1, \psi_2 \in \text{Cl}(\psi) \);
  - if \( \psi' = \langle a \rangle \psi'' \) or \( [a]\psi'' \) for some \( a \in \text{Act} \), then \( \psi'' \in \text{Cl}(\psi) \);
  - if \( \psi' = \sigma X.\psi'' \), then \( \psi''[\sigma X.\psi''/X] \in \text{Cl}(\psi) \), with \( \sigma \) either \( \mu \) or \( \nu \).

Also, we let \( \text{AO}(S) \) represent the set of and-or trees with elements of a set \( S \) as leaves. The core of the model checking algorithm is the construction of a dependency graph \( \text{Dg}(s, \psi) \), to compute \( \text{Pr}_{L,s}^\text{max}(\psi) \), such that all the formulae appearing in the graph will be in the set \( \text{AO}(\text{Cl}(\psi)) \). When constructing a
The sets $N \times N$ formulae of $PE$ valued one $\psi \in \{\psi\}$ formula. Otherwise, every leaf of $AO(\psi)$ is in the action form, $\langle a\rangle\psi'$, and no action may guard more than one leaf.

Given a state $s$, a formula $\psi'$ can be transformed into a semantically equivalent one $\psi''$ that is in factored form $[^1]$ as: $\psi'' = \text{GRP}(PE(s, FPE(\psi')))$. $PE(s, \psi')$ partially evaluates $\psi'$, by evaluating unguarded non-probabilistic subformulae of $\psi'$ as well as all unguarded modal subformulae with actions absent at state $s$, yielding $\texttt{tt}$ or $\texttt{ff}$ for each, and simplifying the result. $[^2]$ Then $((s, \psi'), \varepsilon, (s, \psi'')) \in E$.

**Definition 16** (Factored form). A factored formula $\psi$ can be trivial, when $\psi \in \{\texttt{tt}, \texttt{ff}\}$. Otherwise, every leaf of $AO(\psi)$ is in the action form, $\langle a\rangle\psi'$, and no action may guard more than one leaf.

**Definition 17** (Dependency graph). The dependency graph for model checking a formula $\psi$ with respect to a state $s$ in PLTS $L$, denoted by $Dg(s, \psi)$, is a directed graph $(N, E)$, where node set $N \subseteq S \times AO(Cl(\psi))$, and edge set $E \subseteq N \times (Act \cup \{\varepsilon, \varepsilon^\land, \varepsilon^\lor\}) \times N$; i.e., the edges are labeled from $Act \cup \{\varepsilon, \varepsilon^\land, \varepsilon^\lor\}$. The sets $N$ and $E$ are the smallest such that:

- $(s, \psi) \in N$.

- If $(s', \psi') \in N$, $\psi'$ is not in factored form: if equivalent $\psi''$ in factored form exists, then $(s', \psi'') \in N$ and $((s', \psi'), \varepsilon, (s', \psi'')) \in E$.

- If $(s', \psi'_i \oplus \psi'_j) \in N$, then $(s', \psi'_i) \in N$ for $i = 1, 2$. Moreover, $((s', \psi'_i \oplus \psi'_j), \varepsilon, (s', \psi'_i)) \in E$ for $i = 1, 2$, and $\oplus \in \{\land, \lor\}$.

- If $(s', \langle a\rangle\psi') \in N$, then $(s'', \psi') \in N$ for each $s''$ such that $(s', a, s'') \in \delta$. Moreover, $((s', \langle a\rangle\psi'), a, (s'', \psi')) \in E$.

If $(s', \psi') \in N$ and $\psi'$ has no factored form, then the dependency graph construction fails.

When we transform $\psi'$ to the factored form $\psi''$, the semantics does not change, i.e., $\Theta_{L,s'}(\psi') = \Theta_{L,s'}(\psi'')$. For the factored formulae, standard XPL semantics apply (Table 1). Note that we can assume action nodes to be of the form $(s', \langle a\rangle\psi')$, as the action $a$ must then be present at state $s'$. From these semantics, we also get the relationships for the probabilistic values. Here, $\prod$ is the standard product operator, while $\prod_{i \in I} x_i = 1 - \prod_{i \in I}(1 - x_i)$.

**Lemma 18** (Probabilistic values). Fix $Dg(s_0, \psi) = (N, E)$. The probabilistic value $\Pr_{L,s}^{\text{max}}(\psi)$ for a node $(s, \psi')$ is as follows:

- $\Pr_{L,s}^{\text{max}}(\texttt{ff}) = 0$ and $\Pr_{L,s}^{\text{max}}(\texttt{tt}) = 1$.

---

$[^1]$ We may use the DNF version of $\psi'$ to check for equivalence with existing nodes, but not for finding the factored form.

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

12
• If \((s, \psi')\) is an and-node, then:
  \[
  \Pr_{L,s}^{\max}(\psi') = \prod_{(s, \psi'),(s, \psi')_i) \in E} \Pr_{L,s}^{\max}(\psi'_{i}).
  \]

• If \((s, \psi')\) is an or-node, then:
  \[
  \Pr_{L,s}^{\max}(\psi') = \coprod_{(s, \psi'),(s, \psi')_i) \in E} \Pr_{L,s}^{\max}(\psi'_{i}).
  \]

• If \((s, \psi')\) is an action node, i.e., \(\psi' = \langle a \rangle \psi'_a\), then:
  \[
  \Pr_{L,s}^{\max}(\psi') = \max_{c \in N} \sum_{(s, \psi'),(s', \psi'_a) \in E} P(s, a, s', c) \cdot \Pr_{L,s'}^{\max}(\psi'_{a}).
  \]

• The remaining nodes \((s, \psi')\) have a unique successor \((s, \psi'')\) with \(\Pr_{L,s}^{\max}(\psi') = \Pr_{L,s}^{\max}(\psi'').\)

Proof. Most of the cases are straightforward and similar to the GPL model checking algorithm [1, Lemma 8] and a result for two-player stochastic parity games [2, 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 is as in GPL and [4]; we explain the nondeterministic choice. A PLTS scheduler makes a choice for an action given the partial computation \(\sigma\). Here, this choice is made based on a formula, \(\psi'_a\), to be satisfied. When the initial formula \(\psi\) is separable, this is well-defined: given \(L, s\), and \(\psi\), the scheduler can deduce \(\psi'_a\) from \(\sigma\), a la traversal of the dependency graph.

We note that, although a particular choice may maximize \(\Pr_{L,s}^{\max}(\psi')\), 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 \(\epsilon\)-optimal schedulers for any \(\epsilon > 0\) [5, 2]. The probabilistic value may be predicated on making a different choice eventually. The formulation in Lemma 18 is consistent with this possibility, and the existence of \(\epsilon\)-optimal schedulers may be justified through a common method, called strategy improvement or strategy stealing [4, 2]. The intuition is that, in case of a loop, we can add a choice to succeed immediately with the maximum probability for the state. This cannot increase the probability, and the maximizing scheduler can otherwise be the same, if this choice does not arise.

Theorem 19 (Model checking termination). The graph construction of \(Dg(s, \psi)\) terminates for any XPL formula \(\psi\) and PLTS \(L\). Moreover, if \(\psi\) is separable, the XPL model checking algorithm will complete the construction of the dependency graph.

Proof. \(Cl(\psi)\) is finite, so \(AO(Cl(\psi))\) (for DNF versions used for equivalence checking) is finite. The number of actions in \(L\) and \(\psi\) is finite, so the number of factored formulae is finite. This is sufficient to guarantee termination, as we fail when we cannot construct a factored formula. Meanwhile, separability of \(\psi\) implies that we can construct a factored formula from any \(\psi' \in AO(Cl(\psi))\).
Our primary contribution is the completed dependency graph for a separable formula $\psi$. For model checking separable XPL formulae, we show how, given the graph, to compare the probabilistic value of $\psi$ at a state $s$ against a threshold $p$. We do this by first constructing a system of polynomial min/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 $\leq 1$; or
- of the form $\min(V')$ or $\max(V')$ where $V' \subseteq V$.

Furthermore, the equations form a stratified system, where each variable $x_i$ can be assigned a stratum $j = \text{stratum}(x_i)$ such that $x_i$ is defined in terms of only variables of the form $x_k$ such that $\text{stratum}(x_k) \leq \text{stratum}(x_i)$ (cf. [10, Def. 9]); and variables in the same stratum $j$ fall under the same fixed point.

**Theorem 20.** Given a real value $p$, a system of polynomial min/max fixed point equations and a distinguished variable $x$ defined in the system, whether or not $x > p$ (or $x \geq p$) in its solution is decidable.

**Proof.** We write the min/max polynomial system, $x = P(x)$, as a sentence in the first-order theory of real closed fields, similar to [10]. The additional comparison will be $x_0 > p$ or $x_0 \geq p$. Along with the equation system, we need to encode fixed points and min/max.

We can encode $x_i = \max(x_j, x_k)$ as (13) (cf. [6, Section 5]):

$$x_i \geq x_j \land x_i \geq x_k \land (x_i \leq x_j \lor x_i \leq x_k).$$

(13)

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 (14):

$$\forall x'_I. \left( \bigwedge_{i \in I} x'_i = P_i(x'_I, x_{V \setminus I}) \Rightarrow \bigwedge_{i \in I} x_i \leq x'_i \right).$$

(14)

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 $x > p$ (or $x \geq p$), [13]-[14], and their counterparts encoding min and greatest fixed point, are sentences in a first order theory of real closed fields, which is decidable [11]. Hence the decidability of $x > p$ (and $x \geq p$) in the solution to the fixed point equations follows.

We use the above result to determine whether or not $\Pr_{L,s}^{\max}(\psi) > p$ (similarly, $\geq p$) for a separable XPL formula $\psi$. The polynomial fixed point system is derived similarly to [1, Section 4.1.2], with a variable $x_{(s,\psi)}$ for each node $(s, \psi)$ in the dependency graph $Dg(s, \psi)$, and equations based on Lemma 18.
If \( \psi \) is not in factored form, then \((s, \psi)\) has a unique edge labeled by \( \varepsilon \) to a node \((s, \psi')\), and \( x(s, \psi) = x(s, \psi') \).

- \( x(s, \varnothing) = 0 \) and \( x(s, \top) = 1 \).

- If \((s, \psi)\) is an and-node, then \( x(s, \psi) = \prod_{((s, \psi_1), \varepsilon, (s, \psi_i)) \in E} x(s, \psi_i) \).

- If \((s, \psi)\) is an or-node, then \( x(s, \psi) = \bigvee_{((s, \psi), \varepsilon, (s, \psi_i)) \in E} x(s, \psi_i) \).

- If \((s, \psi)\) is an action node and \( \psi = \langle a \rangle \psi' \), then
  \[
  x(s, \psi) = \max_{c \in \mathbb{N}} \sum_{((s, \psi_1), a, (s', \psi_a)) \in E} P(s, a, s', c) \cdot x(s', \psi_a).
  \]

**Theorem 21** (Correctness). The construction of the dependency graph \( D_g(s, \psi) \), when \( \psi \) is separable, yields a polynomial min/max fixed point equation system, such that the value of \( x(s, \psi) \) in its solution is \( \Pr_{L,s}^{\max}(\psi) \).

**Proof.** The correctness result follows from Lemma 18 and the semantics of fixed points given by Equation 14 (and its counterpart). \( \square \)

Along the same lines as for \( \Pr_{L,s}^{\max} \), we can construct a polynomial min/max fixed point equation system whose solution is \( \Pr_{L,s}^{\min}(\psi) \).

Consequently, we have:

**Corollary 22** (Decidability). Given a state formula \( \varphi \) with separable subformulas, a PLTS \( L \) and a state \( s \) in \( L \), whether or not \( s \models_L \varphi \) is decidable.

**Example 23** (Model Checking). For the PLTS \( L \) in Fig. [1a] and fuzzy formula \( \psi = \mu X.[a][b]X \land [a][c]X \), we have symmetric nondeterministic 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 \( \psi_{bc} = [b]\psi \land [c]\psi \), we get the dependency graph shown in Fig. [2].

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

\[
\begin{align*}
x_1^a &= x_2^b c \\
x_2^b &= x_3^a \cdot x_5^a \\
x_3^a &= \frac{1}{2} x_6^b c + \frac{2}{3} x_2^b c \\
x_4^a &= \frac{1}{2} x_6^b c + \frac{1}{3} x_2^b c \\
x_5^b &= 1 \\
x_6^b &= 1
\end{align*}
\]

(15)

Solving the equations, we get \( \Pr_{L,s_1}^{\max}(\psi) = x_1^a = \frac{1}{4} \).

Note that the model checking algorithm can be broken into the following two parts: writing down a polynomial system, and then finding the (approximate) solution. The first part is bounded double-exponentially in the size of the fuzzy formula, as we deal with (for complexity purposes on equivalence checking) DNF formulae from the Fisher-Ladner closure. For the second part, value
iteration is guaranteed to converge, when computing a single least or greatest fixed point [5], but may be exponentially slow in the number of digits of precision [12]. For polynomial systems arising from conjunctive formulae, alternative approximation methods have been proven to be efficient [13].

5 Encoding Other Model Checking Problems

5.1 Model Checking PCTL* over MDPs

PCTL* is a widely used and well-known logic for specifying properties over Markov chains and MDPs. The syntax of PCTL* may be given as follows, where \( A \in \text{Prop} \) and \( \phi \) and \( \psi \) represent state formulae and path formulae, respectively:

\[
\begin{align*}
\phi & ::= A \mid \phi \land \phi \mid \neg \phi \mid \Pr_{>p} \psi \mid \Pr_{\geq p} \psi \\
\psi & ::= \phi \mid X\psi \mid \psi U \psi \mid \psi \land \psi \mid \neg \psi
\end{align*}
\]

This is similar to the syntax given by [4, Chapter 9], except omitting the bounded \( \text{until} \) operator.

In [1] Sect. 3.2, PCTL* model checking over Markov chains was encoded in terms of GPL model checking over RPLTSs. First of all, Markov chains are represented as RPLTSs with one action label (\( \text{Act} = \{a\} \)). Due to this, all d-trees are paths representing runs in the Markov chain. Thus the linear-time
Table 4: Encoding of PCTL* over MDPs

\[
E_{PCTL^*}(\gamma) = \begin{cases} 
\gamma, & \gamma \in \text{Prop}, \\
\neg(E_{PCTL^*}(\gamma')), & \gamma = \neg\gamma' \\
E_{PCTL^*}(\gamma_1) \land E_{PCTL^*}(\gamma_2), & \gamma = \gamma_1 \land \gamma_2 \\
\Pr_{\ge p} E_{PCTL^*}(\psi), & \gamma = \Pr_{\ge p} \psi \\
\langle a \rangle E_{PCTL^*}(\psi), & \gamma = X\psi \\
\mu X E_{PCTL^*}(\psi_2) \lor (E_{PCTL^*}(\psi_1) \land \langle a \rangle X), & \gamma = \psi_1 U \psi_2.
\end{cases}
\]

The semantics of PCTL* carries over, since all the “trees” degenerate to paths. The GPL encoding of PCTL* then relies on the following three basic steps:

1. Next state operator X is encoded in terms of the diamond modality \( \langle a \rangle \) in GPL.
2. Until formulae \( \psi_1 U \psi_2 \) are encoded by unrolling them as \( \psi_2 \lor (\psi_1 \land X(\psi_1 U \psi_2)) \) and using a least fixed point GPL formula to represent the unrolling.
3. Formulae with negation of the form \( \neg \psi \) are encoded by finding negating the encoding of \( \psi \).

Other operators including PCTL* path quantifiers have corresponding operators in GPL, and are translated directly.

The above encoding has a significant limitation: although RPLTSs in general can exhibit both nondeterministic and probabilistic choices, the GPL-based encoding was only for model checking PCTL* only over Markov chains, and not over MDPs. This is because the nondeterminism in MDPs has linear-time semantics, while all nondeterminism in GPL is under the branching-time semantics. Indeed, this nondeterminism is entirely unused in the above encoding by limiting RPLTSs to one action label \( \langle \text{Act} = \{a\} \rangle \), which, in turn made every d-tree into a path.

As XPL semantics for fuzzy formulae is also defined over d-trees, the PCTL* encoding of [1, Sect. 3.2] carries over to XPL essentially unchanged (we use XPL’s \( \Pr_{\max} \) for GPL’s \( \Pr \), since we assume a maximizing scheduler). For model checking MDPs, we represent them as PLTSs, treating the internal nondeterminism among the actions in an MDP as internal nondeterminism in a PLTS as well. Note that d-trees of a PLTS obtained from an MDP are still paths, since branching in the d-trees only represents external nondeterminism. Consequently, the addition of internal nondeterminism in PLTSs, interpreted under the linear-time semantics, is orthogonal to the problem of encoding PCTL*, because the internal nondeterminism is resolved by the time we reach d-trees. Model checking of a PCTL* formula \( \gamma \) over an MDP is cast as XPL model checking of the corresponding PLTS, where the XPL formula is generated by
$E_{PCTL^*}(\gamma)$ defined in Table 4. In the definition, “$\neg(\psi)$” represents the negation of an XPL formula, also expressed in XPL. Note that, as stated earlier, the translation of PCTL* formulae to XPL formulae is virtually identical to the translation to GPL [1, Sect 3.2]. The novelty is that we have identified the XPL formulae resulting from our translation as separable, and hence PCTL* properties can be successfully model checked over MDPs with our XPL model checking algorithm.

5.2 Encoding of RMDP Termination

We consider recursive MDPs (RMDPs) [6] as a nondeterministic extension of Recursive Markov Chains (RMCs) [18]. We discuss a more general model, called recursive simple stochastic games (RSSGs); formally, an RSSG $A$ is a tuple $(A_1, \ldots, A_k)$, where each component graph $A_i$ is a septuple $(N_i, B_i, Y_i, E_{ni}, E_{xi}, \text{pl}_i, \delta_i)$:

- $N_i$ is a set of nodes, containing subsets $E_{ni}$ and $E_{xi}$ of entry and exit nodes, respectively.
- $B_i$ is a set of boxes, with a mapping $Y_i : B_i \rightarrow \{1, \ldots, k\}$ assigning each box to a component. Each box has a set of call and return ports, corresponding to the entry and exit nodes, respectively, in the corresponding components: $\text{Call}_b = \{ (b, en) \mid en \in E_{ni}(b) \}$, $\text{Return}_b = \{ (b, ex) \mid ex \in E_{xi}(b) \}$. Additionally, we have:
  \[
  \text{Call}\,^i = \bigcup_{b \in B_i} \text{Call}_b, \quad \text{Return}\,^i = \bigcup_{b \in B_i} \text{Return}_b, \quad Q_i = N_i \cup \text{Call}\,^i \cup \text{Return}\,^i. 
  \]
- $\text{pl}_i : Q_i \rightarrow \{0, 1, 2\}$ is a mapping that specifies whether, at each state, the choice is probabilistic (i.e., player 0), or nondeterministic (player 1: maximizing, player 2: minimizing). As any $u \in \text{Call}\,^i \cup \text{Ex}_i$ has no outgoing transitions, let $\text{pl}_i(u) = 0$ for these states.
- $\delta_i$ is the transition relation, with transitions of the form $(u, p_{uv}, v)$, when $\text{pl}_i(u) = 0$ 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 $u$, $\sum_{v' : (u, v') \in \delta_i} p_{uv'} = 1$. Meanwhile, the nondeterministic extension yields transitions of the form $(u, \perp, v)$ when $\text{pl}_i(u) > 0$.

Recursive MDPs (RMDPs) only have a player 1 or player 2, depending on whether they are maximizing or minimizing, respectively. Termination probabilities can be computed for 1-RSSGs, and are always achieved, for both players, with a strategy limited to a class called stackless and memoryless (SM) [6]. The essence of SM strategies is that in each nondeterministic choice, the selection...
is fixed to a single state from its distribution, which makes the resolution of the nondeterministic choices substantially simpler than in the general case. For multi-exit RSSGs, the termination probability is determined, although an optimal strategy may not exist, and the problem of computing the probability is undecidable, in general. SM strategies are inadequate even for 2-exit RMDPs [6]. Figure 3 shows a recursive MDP with two components, A and B. Any call to A nondeterministically results in either a call to B (via box $b_1$) or a transition to $u$.

5.2.1 Translating RMDPs to PLTSs

Given an RMDP $A$, we can define a PLTS $L$ that simulates $A$, with $\text{Act} = \{p,n,c,r_i,e_i\}$ and states of the PLTS corresponding to nodes of the RMDP. We retain the RMDPs transitions, labeling them as $n$ for actions from a nondeterministic choice and $p$ for probabilistic choice. To this basic structure we add three new kinds of edges:

- $e_i$ for the $i$th exit node of a component,
- $c$ edges from a call port to the called component’s entry node, and
- $r_i$ edges from a call port to each return port in the box.

While $c$ edges denote control transfer due to a call, $r$ edges summarize returns from the called procedure. Figure 3 shows the result of the translation for one component of the RMDP. Formally, we define the PLTS $L$ as follows:

**Definition 24** (Translated RMDP). The translated RMDP $A$ is a PLTS $L = (S, \delta, P, I)$:

- 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., $S = \bigcup Q_i$. Additionally, we associate a consistent index with each state corresponding to an exit node or a return port.

- The transition relation $\delta$ has all the transitions of the components, labeled by action $p$ for the probabilistic transitions and $n$ for the nondeterministic ones. Thus, when $(u, p_{uv}, v) \in \delta_i$ for any $i$, then $(u, p, v) \in \delta$, and when $(u, \perp, v) \in \delta_i$, $(u, n, v) \in \delta$. Additionally, we have $((b, en), c, en) \in \delta$ and $((b, en), r_i, (b, ex_i)) \in \delta$ for every box $b$, and $(ex_i, e_i, ex_i) \in \delta$ for every exit node. Note the indices used.

- The transition probability distribution $P$ is defined as $P(u, p, v, \cdot) = p_{uv}$ as given for the RMDP $A$, $P(u, n, v, c(v)) = 1$, where $c : S \rightarrow \mathbb{N}$ is a one-to-one function (when $c \neq c(v)$ for any $v$ with $(u, n, v) \in \delta$). $P(u, n, v, c) = 1$ for an arbitrary $v$ with $(u, n, v) \in \delta$, and $P(\cdot) = 1$ if the action is not $p$ or $n$.

- We do not use the interpretation in the translation, i.e., $I(s) = \emptyset$ for any state $s$, unless additional relevant information about the RMDP $A$ is available.
Figure 3: Example RMDP with Call, Return, and Exit edges added to A

For RMCs, the translation yields a simulating RPLTS $L$ (no $n$ actions).

Intuitively, $L$ preserves all the non-recursive transition structure of $A$ via the actions labeled by $p$ and $n$. There are additional $c$ actions to model call transitions. Note that each call port will have a single outgoing $c$ transition, while the entry nodes may have multiple incoming $c$ transitions. Meanwhile, we need a different design to associate exit nodes with return ports, as an exit node may be associated with multiple return ports. Thus, we have indexed $e$ and $r$ actions and require a standard formula to model termination. We note that the resulting structure is similar to the nested state machines (NSM) [14], with the $p/n$, $c$, $r$, and $e$ edges corresponding to the loc (local), call, jump, and ret edges, respectively, in the NSM model.

Termination of 1-RMDPs can be encoded as the following separable formula:

$$\psi_1 = \mu X. (e_1)\text{tt} \lor (p)X \lor (n)X \lor ((c)X \land (r_1)X)$$

Termination of multi-exit RMDPs is undecidable, in general [6]. We can still encode it in XPL, but the resulting formula is not separable: the termination formula for a 2-exit RMDP is entangled on the $c$ action.

$$\psi_2 = \mu X. (e_2)\text{tt} \lor (p)\psi_2 \lor (n)\psi_2 \lor ((c)\psi_1 \land (r_1)\psi_2) \lor ((c)\psi_1 \land (r_2)\psi_2) \lor (e_2)\text{tt} \lor (p)\psi_2 \lor (n)\psi_2 \lor ((c)\psi_1 \land (r_1)\psi_2) \lor ((c)\psi_1 \land (r_2)\psi_2)$$

We note that the conjunction $(c)X \land (r_1)X$ in (16) is independent. Additionally, the disjunction between the two conjuncts in (17) is mutually exclusive, since $(c)\psi_1$ and $(c)\psi_2$ correspond to eventually reaching distinct exits; however, it is correct to sum them only for RMCs, as the nondeterministic choices in RMDPs preclude the simple summation of mutually exclusive outcomes.

5.3 PTTL and Branching Processes

PCTL* [4] may be considered a linear-time logic, in the sense that its fuzzy formulae are essentially full LTL. Similarly, PCTL [10] is not the only plausi-
A similar logic, Probabilistic Tree Temporal Logic (PTTL), has been independently introduced [17]. Branching Processes (BPs) are a branching-time extension of Markov chains, and PTTL is a logic over BPs. The problem of BP extinction corresponds to termination of 1-exit RMCs [18]. BPs have also been extended with nondeterminism, yielding Branching MDPs (BMDPs), for which the extinction and reachability problems have been analyzed [5, 6].

We write the syntax of PTTL [17, Definition 18], where \( A \in \text{Prop} \), and we refer to \( \phi \) and \( \psi \) as state and fuzzy formulae, as for XPL:

\[
\begin{align*}
\phi & ::= A \mid \neg \phi \mid \phi \land \phi \mid \text{Pr}_{>p} \psi \mid \text{Pr}_{\geq p} \psi \\
\psi & ::= AX\phi \mid EX\phi \mid A[\phi U \psi] \mid E[\phi U \psi] \mid A[\phi R \psi] \mid E[\phi R \psi]
\end{align*}
\]

We can translate BPs to RPLTSs and BMDPs to PLTSs, but unlike with RMDPs, we may view PLTSs simply as more general than BMDPs. So, we give the semantics for PTTL over PLTSs (without terminal states), assuming maximizing schedulers, by encoding it in XPL, as \( E_{\text{PTTL}}(\gamma) \), in Table 5. Note that PTTL’s fuzzy formulae are substantially more limited than full CTL, which does, however, place them as a subset of separable XPL. Thus, we have also extended PTTL as a logic over BMDPs, whose properties we can verify with the XPL model checking algorithm.

### Table 5: Encoding of PTTL over BMDPs

| \( E_{\text{PTTL}}(\gamma) \) | \( \gamma \in \text{Prop} \), \( \gamma' \) |
|--------------------------------|---------------------------------|
| \( \text{neg}(E_{\text{PTTL}}(\gamma')) \) | \( \gamma = \neg \gamma' \), \( \gamma' \) |
| \( E_{\text{PTTL}}(\gamma_1) \land E_{\text{PTTL}}(\gamma_2) \) | \( \gamma = \gamma_1 \land \gamma_2 \), \( \gamma_1, \gamma_2 \) |
| \( \text{Pr}_{\geq p} \max E_{\text{PTTL}}(\psi) \) | \( \gamma = \text{Pr}_{\geq p} \psi \), \( \gamma \) |
| \( \text{Pr}_{>p} \max E_{\text{PTTL}}(\psi) \) | \( \gamma = \text{Pr}_{>p} \psi \), \( \gamma \) |
| \( [-\neg \text{PTTL}(\phi)] \) | \( \gamma = AX\phi \), \( \gamma \) |
| \( (\neg) \text{PTTL}(\phi) \) | \( \gamma = EX\phi \), \( \gamma \) |
| \( \mu X.\text{PTTL}(\phi_2) \lor (E_{\text{PTTL}}(\phi_1) \land [-\neg X]) \) | \( \gamma = A[\phi_1 U \phi_2] \), \( \gamma \) |
| \( \mu X.\text{PTTL}(\phi_2) \lor (E_{\text{PTTL}}(\phi_1) \land (\neg) X) \) | \( \gamma = E[\phi_1 U \phi_2] \), \( \gamma \) |
| \( \nu X.\text{PTTL}(\phi_2) \land (E_{\text{PTTL}}(\phi_1) \lor [-\neg X]) \) | \( \gamma = A[\phi_1 R \phi_2] \), \( \gamma \) |
| \( \nu X.\text{PTTL}(\phi_2) \land (E_{\text{PTTL}}(\phi_1) \lor (\neg) X) \) | \( \gamma = E[\phi_1 R \phi_2] \), \( \gamma \) |

### 6 Discussion and Future Work

Previous attempts to extend GPL included allowing systems with internal nondeterminism while still resolving the probabilistic choices first [19], and EGRL,
which had similar syntax and semantics to XPL, but limited the model checking to non-recursive formulae \[20\].

Following GPL, XPL treats conjunction in a traditional manner, retaining the properties that $\psi \land \neg \psi = \mathbf{ff}$, and $\psi \land \psi = \psi$ for any formula $\psi$. However, the probability value of $\psi_1 \land \psi_2$ cannot be computed based on the probability values of the conjuncts $\psi_1$ and $\psi_2$. This makes model checking in XPL more complex, but also contributes to its expressiveness.

Another probabilistic extension of $\mu$-calculus is $\mu\mathbb{L}$. In contrast to XPL, the most expressive version of $\mu\mathbb{L}$, denoted $\mu\mathbb{L}_\oplus \otimes$ \[2, 21\], defines three conjunction operators and their duals such that their probability values can be computed from the probabilities of the conjuncts. The logic $\mu\mathbb{L}_\oplus \otimes$ is able to support branching time and an intuitive game semantics \[2\]. Along the same lines as our XPL encoding, we can encode termination of 1-exit RMDPs as model checking in $\mu\mathbb{L}_\oplus \otimes$, and RMC termination in $\mu\mathbb{L}_\otimes \oplus$. However, attempting to encode multi-exit RMDP termination in $\mu\mathbb{L}_\otimes \oplus$ similarly to multi-exit RMC termination would lead to an incorrect, rather than undecidable, encoding. Determining the relationship between XPL and $\mu\mathbb{L}_\otimes \oplus$ in branching time is an important problem. Other recent probabilistic extensions of $\mu$-calculus include the Łukasiewicz $\mu$-calculus \[22\] and $\mu^\mathbb{P}$-calculus \[23\], which can encode PCTL* over MDPs, and P\muTL \[24\], but all these limit nondeterminism to the linear-time semantics.

Although closely related, algorithms to check properties of RMCs (and pPDSs \[25\]) were developed independently \[18\]. 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 \[3\], but has remained largely unexplored.

There has been significant interest in the study of expressive systems with nondeterministic choices, such as RMDPs and Branching MDP (BMDPs) \[6\]. At the same time, the understanding of the polynomial systems has expanded. In \[26\], 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 \[13\]. While \[18\] 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 RMDP termination and, later, BMDP reachability, both having polynomial-time complexity for min/maxPPSs \[26, 5\].

Systems producing equations in PPS form show an interesting characteristic: that the properties are expressible as purely conjunctive or purely disjunctive formulae. Recall that such formulae are trivially separable. Polynomial systems equivalent to those arising from separable GPL have recently been considered in a more general setting in the context of game automata \[10\], followed by an undecidability result for more general properties on the automata \[27\]. Characterizing equation systems that arise from separable formulae and investigating their efficient solution is an interesting open problem.
References

[1] R. Cleaveland, S. P. Iyer, and M. Narasimha, “Probabilistic temporal logics via the modal mu-calculus,” *Theoretical Computer Science*, vol. 342, no. 2-3, pp. 316–350, 2005.

[2] M. Mio, “Probabilistic modal $\mu$-calculus with independent product,” in *FOSSACS*, ser. LNCS, vol. 6604. Springer, 2011, pp. 290–304.

[3] A. Gorlin, C. R. Ramakrishnan, and S. A. Smolka, “Model checking with probabilistic tabled logic programming,” *TPLP*, vol. 12, no. 4-5, pp. 681–700, 2012.

[4] C. Baier, “On algorithmic verification methods for probabilistic systems,” 1998, habilitation thesis, Fakultät für Mathematik & Informatik, Universität Mannheim.

[5] K. Etessami, A. Stewart, and M. Yannakakis, “Greatest fixed points of probabilistic min/max polynomial equations, and reachability for branching Markov decision processes,” in *ICALP, Part II*. Berlin, Heidelberg: Springer, 2015, pp. 184–196.

[6] K. Etessami and M. Yannakakis, “Recursive Markov decision processes and recursive stochastic games,” *J. ACM*, vol. 62, no. 2, pp. 11:1–11:69, May 2015.

[7] R. Segala, “A compositional trace-based semantics for probabilistic automata,” in *CONCUR*, ser. LNCS, vol. 962. Springer, 1995, pp. 234–248.

[8] R. Segala and A. Turrini, “Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models,” in *QEST*. IEEE Computer Society, 2005, pp. 44–53.

[9] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden, “Weak bisimulation is sound and complete for PCTL*,” in *CONCUR*, ser. LNCS. Springer Berlin Heidelberg, 2002, vol. 2421, pp. 355–370.

[10] H. Michalewski and M. Mio, “On the problem of computing the probability of regular sets of trees,” in *FSTTCS*, ser. LIPIcs, vol. 45. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 489–502.

[11] A. Tarski, “A decision method for elementary algebra and geometry,” *Bulletin of the American Mathematical Society*, vol. 59, 1951.

[12] S. Kiefer, M. Luttenberger, and J. Esparza, “On the convergence of Newton’s method for monotone systems of polynomial equations,” in *STOC*, 2007, pp. 217–226.

[13] K. Etessami, A. Stewart, and M. Yannakakis, “Polynomial time algorithms for multi-type branching processes and stochastic context-free grammars,” in *STOC*. ACM, 2012, pp. 579–588.
[14] R. Alur, S. Chaudhuri, and P. Madhusudan, “Software model checking using languages of nested trees,” ACM Trans. Program. Lang. Syst., vol. 33, no. 5, pp. 15:1–15:45, Nov. 2011.

[15] A. Gorlin and C. R. Ramakrishnan, “XPL: an extended probabilistic logic for probabilistic transition systems,” CoRR, vol. abs/1604.06118, 2016.

[16] H. Hansson and B. Jonsson, “A logic for reasoning about time and reliability,” Formal Aspects of Computing, vol. 6, no. 5, pp. 512–535, 1994.

[17] T. Chen, K. Dräger, and S. Kiefer, “Model checking stochastic branching processes,” in MFCS. Berlin, Heidelberg: Springer, 2012, pp. 271–282.

[18] K. Etessami and M. Yannakakis, “Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations,” J. ACM, vol. 56, no. 1, pp. 1:1–1:66, Feb. 2009.

[19] R. Cleaveland and S. P. Iyer, “Branching time probabilistic model checking,” in ICALP Workshops, vol. 8. Citeseer, 2000, pp. 487–500.

[20] A. Soni, “Probabilistic and nondeterministic systems,” 2004, masters thesis, North Carolina State University.

[21] M. Mio, “Game semantics for probabilistic modal mu-calculi,” 2012, PhD thesis, The University of Edinburgh.

[22] M. Mio and A. Simpson, “Łukasiewicz mu-calculus,” in FICS, ser. EPTCS, vol. 126, 2013, pp. 87–104.

[23] P. Castro, C. Kilmurray, and N. Piterman, “Tractable probabilistic mu-calculus that expresses probabilistic temporal logics,” in STACS, ser. LIPIcs, vol. 30. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 211–223.

[24] W. Liu, L. Song, J. Wang, and L. Zhang, “A simple probabilistic extension of modal mu-calculus,” in IJCAI, 2015.

[25] J. Esparza, A. Kucera, and R. Mayr, “Model checking probabilistic pushdown automata,” in LICS, 2004, pp. 12–21.

[26] K. Etessami, A. Stewart, and M. Yannakakis, “Polynomial time algorithms for branching Markov decision processes and probabilistic min(max) polynomial Bellman equations,” in ICALP, Part I. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 314–326.

[27] M. Przybylko and M. Skrzypeczak, “On the complexity of branching games with regular conditions,” in LIPIcs, vol. 58. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.