

Probabilistic Temporal Logics via the Modal Mu-Calculus

Rance Cleaveland^a S. Purushothaman Iyer^b
Murali Narasimha^c

^a *Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY
11794-4400, USA, rance@cs.sunysb.edu*

^b *Department of Computer Science, North Carolina State University, Raleigh, NC
27695, USA, purush@csc.ncsu.edu*

^c *Wireless Research and Development, Ericsson, Research Triangle Park, NC
27709 eusmuna@rtp.ericsson.se*

Abstract

This paper presents a mu-calculus-based modal logic for describing properties of reactive probabilistic labeled transition systems (RPLTSs) and develops a model-checking algorithm for determining whether or not states in finite-state RPLTSs satisfy formulas in the logic. The logic is based on the distinction between (probabilistic) “systems” and (nonprobabilistic) “observations”: using the modal mu-calculus, one may specify sets of observations, and the semantics of our logic then enable statements to be made about the measures of such sets at various system states. The logic may be used to encode a variety of probabilistic modal and temporal logics; in addition, the model-checking problem for it may be reduced to the calculation of solutions to systems of non-linear equations. Finally, the logic induces an equivalence on RPLTSs that coincides with accepted notions of probabilistic bisimulation in the literature.

Key words: Probabilistic transition systems, reactive systems, probabilistic temporal logic, model-checking, probabilistic bisimulation.

* Research supported by ARO under grants DAAG55-98-1-0309, DAAD190110003, DAAD1901100198, and DAAD19-01-1-0683 and by NSF under grants CCR-0098037 and CCR-9988489.

1 Introduction

Classical temporal-logic model checking [CES86,EC80,McM93,QS82] provides an array of techniques for automatically checking the correctness of finite-state systems such as hardware designs and communication protocols. In the model-checking framework, systems are modeled as *transition systems*, and requirements are given as formulae in temporal logic. A model checker accepts two inputs, a transition system and a temporal formula, and returns “true” if the system satisfies the formula and “false” otherwise. Many tools also return diagnostic information if a “false” result is detected.

Traditional transition systems include information about the possible choices of execution steps the system being modeled may engage in in any given system state. Popular temporal logics such as CTL [CES86] and CTL* [EH86] then combine a language for describing properties of system “runs” with quantifiers for indicating when all/some of the runs of a system have a given property. Underlying these logics is an extremely expressive, albeit low-level, logic, the *modal mu-calculus* [Koz83,EL86], that supports the formulation of recursive system properties. Because of its expressiveness, model-checking algorithms have been extensively investigated for the mu-calculus and various of its fragments [And94,BC96,CS93,EL86,LBCJ94].

When system models include *probabilistic* information regarding their execution choices, however, one frequently wishes to determine not just whether or not all/some system behaviors have a given property, but “how many” of them do. Many important questions of design and performance in distributed systems and communication protocols, such as “hot-spot” detection or reliability information, can be addressed more appropriately in such a probabilistic framework. These motivations have led to the study of numerous probabilistic variants of temporal logic and model checking, including [ASB⁺95,BdA95,CY88,Han94] and [HK97,LS91,PZ93,Var85]. However, no unifying mu-calculus-like logic has yet been identified for these probabilistic models.

Another trend in the study of traditional transition systems involves the study of refined notions of choice among execution steps. The process algebra community, in particular, has advocated a distinction between *internal choice* (the system makes the selection) and *external choice* (the environment makes the selection) [BPS01]. The former is sometimes referred to as demonic choice, and the latter angelic choice. The distinctions in these notions are captured semantically via behavioral equivalences such as bisimulation equivalence [Mil89], and failures / testing equivalence [BHR84,DH83]. Probabilistic versions of these theories have been developed by mixing in notions of probabilistic choice to the others, leading to various forms of probabilistic labeled transition sys-

tems [vGSST90a,LS91,SL95]. Noteworthy among these models is the *reactive* model [LS91], in which internal choices are replaced by probabilistic ones, as they represent the first model for which a probabilistic notion of bisimulation has been defined. A mu-calculus-like logic has also been developed that is shown to characterize this notion of equivalence in the sense that equivalent systems satisfy precisely the same properties. The logic, however, suffers from deficiencies as a specification logic, as it is not capable of encoding other probabilistic logics even when augmented with operators for recursive definition.

The goal of this paper is the development of a unifying logic for temporal reasoning about probabilistic systems that is analogous to the role played by the mu-calculus for nondeterministic systems. The work reported here is part of a larger program aimed at a unified framework of logic-based reasoning techniques that cater to specifications containing arbitrary mixtures of probabilistic, external and internal choice. In this paper we restrict our attention to system specifications containing probabilistic and external choice, and develop the following:

- A tree-based semantics for reactive probabilistic labeled transition systems (RPLTS) [vGSST90a,LS91] and a Generalized Probabilistic Logic (GPL) based on the modal mu-calculus [Koz83,EL86] interpreted with respect to the semantics.
- Expressiveness results showing that the logic and the semantics are a conservative extension of traditional treatments of Markov processes and can encode a variety of other probabilistic logics found in the literature.
- An algorithm to model-check RPLTS specifications against GPL formulae.
- A proof that the semantic equality induced by GPL coincides with probabilistic bisimulation [LS91,SL95] for RPLTS.

Given that the two notions of bisimulation for probabilistic systems proposed in the literature [LS91,SL95] coincide for Reactive systems [BS01] our logical characterization is indeed tight; this in turn attests to the correctness of our semantics.

Related work: Related work in extending finite state verification methods to finite state probabilistic systems can be divided into four categories depending upon the systems they consider: (a) purely probabilistic (Markov chains), (b) combination of probabilistic and internal choice (Markov decision processes), (c) probabilistic and external choice (RPLTS), and (d) all three forms of choices (Probabilistic Transition Systems).

Vardi [Var85] addresses the problem of showing when Markov chains and Markov decision processes satisfied LTL formulae with probability 1; these results were extended to all probabilities $p \leq 1$ by Courcoubetis and Yan-

nakakis [CY88]. Several branching time logics and attendant semantics with respect to Markov chains and Markov decision processes have been proposed by Alur, Courcoubetis and Dill [ACD91], by Aziz *et al* [ASB⁺95], and by Bianca and de Alfaro [BdA95].

Work on logics for Reactive systems have resulted in a probabilistic version of Hennessey-Milner logic in [LS91] and in a logic based on mu-calculus in [HK97]. Finally, work on systems that contain all three kinds of choice have resulted in several notions of bisimulation [SL95,BS01].

Outline of the paper: In Section 2 we propose our semantics and our logic. In Section 3 we present our expressiveness results. In Section 4 we provide details of our model-checking algorithm. In Section 5 we prove that our logic provides another characterization of probabilistic bisimulation for RPLTS. We finish with some comments about the utility of our semantics in Section 6.

2 Probabilistic Transition Systems and Generalized Probabilistic Logic

This section introduces the model of probabilistic computation used in this paper and defines the syntax and semantics of our logic, Generalized Probabilistic Logic (GPL).

2.1 Reactive Probabilistic Labeled Transition Systems

We use the *reactive probabilistic labeled transition systems* (RPLTS for short) of [vGSST90b,LS91] as our model of probabilistic computation. These are defined with respect to fixed sets *Act* and *Prop* of atomic *actions* and *propositions*¹, respectively. The former set records the interactions the system may engage in with its environment, while the latter provides information about the states the system may enter.

Definition 1 *A reactive probabilistic labeled transition system L is a tuple (S, δ, P, I) , where*

- S is a countable set of states ranged over by $s, s', s_1 \dots$;
- $\delta \subseteq S \times Act \times S$ is the transition relation;
- $P : \delta \rightarrow (0, 1]$, the transition probability distribution, satisfies:

¹ Definitions of Reactive systems, such as in [LS91], do not include atomic propositions labeling states.

- $\forall s \in S. \forall a \in Act. \sum_{s':(s,a,s') \in \delta} P(s, a, s') \in \{0, 1\}$, and
- $\forall s \in X. \forall a \in Act. (\exists s'. (s, a, s') \in \delta) \Rightarrow \sum_{s':(s,a,s') \in \delta} P(s, a, s') = 1$.
- $I : S \rightarrow 2^{Prop}$ is the interpretation, which records the set of propositions true at a state.

Intuitively, a RPLTS encodes the operational behavior of a system, with S representing the possible system states and δ the execution steps enabled in different system states. A RPLTS in a state s responds to an action a enabled by the environment by probabilistically choosing one of the a -labeled transitions available at s . The quantity $P(s, a, s')$ represents the probability with which the transition (s, a, s') is selected as opposed to other transitions labeled by a emanating from state s . Note that the conditions on P ensure that if $(s, a, s') \in \delta$ for some s' , then $\sum_{s':(s,a,s') \in \delta} P(s, a, s') = 1$. In what follows we write $s \xrightarrow{a}_\delta s'$ if $(s, a, s') \in \delta$; if δ is clear from context then we often use $s \xrightarrow{a} s'$ instead.

In this paper we wish to view a (state in a) RPLTS as an “experiment” in the probabilistic sense, with an “outcome”, or “observation”, representing a resolution of all the possible probabilistic choices of transitions the system might experience as it executes. Such a view is consistent with the definition of Markov chains where a state is viewed as the beginning of an infinite sequence of probabilistic choices [KSK66]. Consider the prototypical example of infinite coin-tossing experiment. Probabilities in this context measures the fraction of infinite sequences that satisfy some condition (such as thrice as many heads as tails). As we are dealing with external non-determinism we are interested in measuring the result of the choices (i.e., button pushes, in the parlance of Milner [Mil89]) made by an external user of a reactive system. More specifically, given a state in the RPLTS we can unroll the RPLTS into an infinite tree rooted at this state. An observation would then be obtained from this tree by resolving all probabilistic choices, i.e. by removing all but one edge for any given action from each node in the tree. This is in keeping with the fact that RPLTSs model external as opposed to internal non-determinism, and is a generalization of the notion of infinite sequences of probabilistic choices in the theory of Markov chains. Figure 1 presents a sample RPLTS, its unrolling from a given state, and an associated observation.

2.1.1 RPLTSs and Measure Spaces of Observations

To define the observation trees of a RPLTS we introduce *partial computations*, which will be used to characterize the nodes of the trees.

Definition 2 Let $L = (S, \delta, P, I)$ be a RPLTS. Then a sequence of the form $s_0 \xrightarrow{a_1} s_1 \cdots \xrightarrow{a_n} s_n$ is a partial computation of L if $n \geq 0$ and for all $0 \leq i < n$, $s_i \xrightarrow{a_{i+1}} s_{i+1}$.

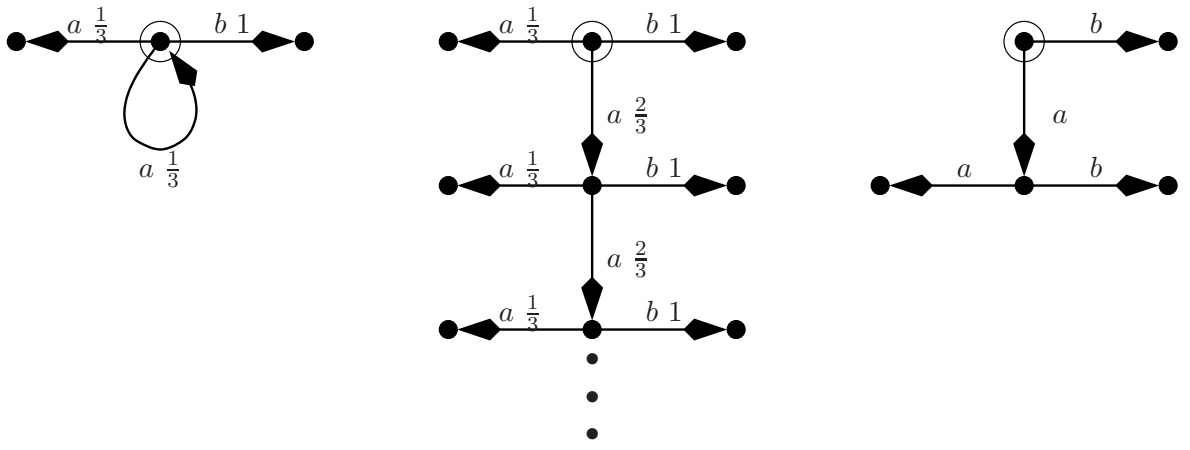


Fig. 1. A RPLTS, its unrolling from a state, and an observation.

Note that any $s \in S$ is a partial computation. If $\sigma = s_0 \xrightarrow{a_1} s_1 \cdots \xrightarrow{a_n} s_n$ is a partial computation then we define $\text{fst}(\sigma)$ to be s_0 and $\text{last}(\sigma)$ to be s_n . We also use \mathcal{C}_L (ranged over by σ, σ', \dots) to refer to the set of all partial computations of L and take $\mathcal{C}_L(s) = \{\sigma \in \mathcal{C}_L \mid \text{fst}(\sigma) = s\}$ for $s \in S$. We define the following notations for partial computations.

Definition 3 Let $\sigma = s_0 \xrightarrow{a_1} s_1 \cdots \xrightarrow{a_n} s_n$ and $\sigma' = s'_0 \xrightarrow{b_1} s'_1 \cdots \xrightarrow{b_{n'}} s'_{n'}$ be partial computations of RPLTS $L = (S, \delta, P, I)$, and let $a \in \text{Act}$.

- (1) If $s_n \xrightarrow{a} s'_0$ then $\sigma \xrightarrow{a} \sigma'$ is the partial computation $s_0 \xrightarrow{a_1} s_1 \cdots \xrightarrow{a_n} s_n \xrightarrow{a} s'_0 \xrightarrow{b_1} s'_1 \cdots \xrightarrow{b_{n'}} s'_{n'}$.
- (2) σ' is a prefix of σ if $\sigma' = s_0 \xrightarrow{a_1} s_1 \cdots \xrightarrow{a_i} s_i$ for some $i \leq n$.

We now introduce the following terminology for sets of partial computations, which will be used to characterize trees.

Definition 4 Let $L = (S, \delta, P, I)$ be a RPLTS, and let $T \subseteq \mathcal{C}_L$ be a set of computations.

- (1) T is prefix-closed if, for every $\sigma \in T$ and σ' a prefix of σ , $\sigma' \in T$.
- (2) T is deterministic if for every $\sigma, \sigma' \in T$ with $\sigma = s_0 \xrightarrow{a_1} s_1 \cdots \xrightarrow{a_n} s_n \xrightarrow{a} s \cdots$ and $\sigma' = s_0 \xrightarrow{a'_1} s'_1 \cdots \xrightarrow{a'_n} s'_n \xrightarrow{a'} s' \cdots$, either $a \neq a'$ or $s = s'$.

The term prefix-closed is standard, but the notion of determinacy of sets of partial computations deserves some comment. Intuitively, if two computations in a deterministic set of partial computations share a common prefix, then the first difference they can exhibit must involve transitions labeled by different actions; they cannot involve different transitions with the same action label.

We can now define the deterministic trees, or *d-trees*, of a RPLTS L as follows.

Definition 5 Let $L = (S, \delta, P, I)$ be a RPLTS. Then $\emptyset \neq T \subseteq \mathcal{C}_L$ is a d-tree

if the following hold.

- (1) There exists an $s \in S$ such that $T \subseteq \mathcal{C}_L(s)$.
- (2) T is prefix-closed.
- (3) T is deterministic.

If T is a d-tree then we use $\text{root}(T)$ to refer to the s such that $T \subseteq \mathcal{C}_L(s)$ and $\text{edges}(T)$ to refer to the relation $\{(\sigma, a, \sigma') \mid \sigma, \sigma' \in T \wedge \exists s' \in S. \sigma' = \sigma \xrightarrow{a} s'\}$.

We use \mathcal{T}_L to refer to all the d-trees of L and set $\mathcal{T}_L(s) = \{T \in \mathcal{T}_L \mid \text{root}(T) = s\}$. We call T' a *prefix* of T if $T' \subseteq T$. We write $T \xrightarrow{a} T'$ provided $T' = \{\sigma \mid \text{root}(T) \xrightarrow{a} \sigma \in T\}$; intuitively, T' is then the subtree of T pointed to by an a -labeled edge. A d-tree T is *finite* if $|T| < \infty$. Finally, we say that a d-tree is *maximal* if there exists no d-tree T' with $T \subset T'$ and use \mathcal{M}_L and $\mathcal{M}_L(s)$ to refer to the set of all maximal d-trees of L and all maximal d-trees of L rooted at s , respectively.

We wish to view the maximal deterministic d-trees of a RPLTS as the “outcomes” (synonymous with observations) of the RPLTS and to talk about the likelihoods of different sets of outcomes. In order to do this, we define a probability space over maximal d-trees rooted at a given state of L . The construction of this space is very similar in spirit to the standard sequence space construction for Markov chains [KSK66]: we define a collection of “basic cylindrical sets” of maximal trees and use them to build a probability space over sets of maximal trees. The technical details appear below; in what follows, fix $L = (S, \delta, P, I)$.

Definition 6 A basic cylindrical subset of $\mathcal{M}_L(s)$ contains all trees sharing a given finite prefix. Let $s \in S$, and let $T \in \mathcal{T}_L(s)$ be finite. Then $B_T \subseteq \mathcal{M}_L(s)$, a basic cylindrical set with common prefix T , is defined as: $B_T = \{T' \in \mathcal{M}_L \mid T \subseteq T'\}$.

We can also define the *measure* of a basic cylindrical set as follows.

Definition 7 Let $T \in \mathcal{T}_L(s)$ be finite, and let B_T be the associated basic cylindrical set. Then the measure, $\mathbf{m}(B_T)$, of B_T is given by:

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

viz. the product of the probabilities of the edges in the finite tree T .

Intuitively, $\mathbf{m}(B_T)$ represents the fraction of all maximal d-trees which have T as a prefix.

For any given state s in L we can form the associated collection of basic cylindrical sets \mathcal{B}_s^- consisting of sets of the form B_T for finite T with $\text{root}(T) = s$. We can then define a probability space $(\mathcal{M}_L(s), \mathcal{B}_s, \mathbf{m}_s)$ as follows.

Definition 8 Let $s \in S$. Then \mathcal{B}_s is the smallest field of sets containing \mathcal{B}_s^- and closed with respect to denumerable unions and complementation. $\mathbf{m}_s : \mathcal{B}_s \rightarrow [0, 1]$ is then defined to be unique extension that satisfies the following conditions:

$$\begin{aligned} \mathbf{m}_s(B_T) &= \mathbf{m}(B_T) \\ \mathbf{m}_s\left(\bigcup_{i \in I} B_i\right) &= \sum_{i \in I} \mathbf{m}_s(B_i) \text{ for pairwise disjoint } B_i \\ \mathbf{m}_s(B^c) &= 1 - \mathbf{m}_s(B) \end{aligned}$$

It is easy to show that for any s , \mathbf{m}_s is a probability measure over \mathcal{B}_s . Consequently, $(\mathcal{M}_L(s), \mathcal{B}_s, \mathbf{m}_s)$ is indeed a probability space. We refer to a set $M \subseteq \mathcal{M}_L(s)$ as *measurable* if $M \in \mathcal{B}_s$.

2.2 Syntax of GPL

Generalized Probabilistic Logic (GPL) is parameterized with respect to a set *Var* of propositional variables (ranged over by $X, Y \dots$), a set *Act* of actions (ranged over by a, b, \dots), and a set *Prop* of atomic propositions (ranged over A ; it is assumed that *Prop* and *Var* are disjoint). The syntax of GPL may then be given using the following BNF-like grammar, where $0 \leq p \leq 1$.

$$\begin{aligned} \phi &::= A \mid \neg A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \text{Pr}_{>p}\psi \mid \text{Pr}_{\geq p}\psi \\ \psi &::= \phi \mid X \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \langle a \rangle \psi \mid [a]\psi \mid \mu X.\psi \mid \nu X.\psi \end{aligned}$$

The operators μ and ν bind variables in the usual sense, and one may define the standard notions of free and bound variables. Also, we refer to an occurrence of a bound variable X in a formula as a μ -occurrence if the closest enclosing binding operator for X is μ and as a ν -occurrence otherwise. GPL formulae are required to satisfy the following additional restrictions:

- in formulae of the form $\text{Pr}_{\geq p}\psi$ or $\text{Pr}_{>p}\psi$ the formula ψ can not contain any free variables, and
- no subformula of the form $\mu X.\psi$ ($\nu X.\psi$) may contain a free ν -occurrence (μ -occurrence) of a variable.²

In what follows we refer to formulae generated from nonterminal ϕ as *state formulae* and those generated from ψ as *fuzzy formulae*; the formulae of GPL are the state formulae. We use $(\phi, \phi' \in) \Phi$ to represent the set of all state

² In other words, formulae must be alternation-free in the sense of [EL86].

formulae and $(\psi, \psi' \in) \Psi$ for the set of all fuzzy formulae. In the remainder of the paper we write $\gamma[\gamma'/X]$ to denote the simultaneous substitution of γ' for all free occurrences of X in γ . We also note that although the logic limits the application of \neg to atomic propositions, this does not restrict the expressiveness of the logic, as we indicate later.

The next subsection defines the formal semantics of GPL, but the intuitive meanings of the operators may be understood as follows. Fuzzy formulae are to be interpreted as specifying sets of *observations* of RPLTSs, which are themselves nonprobabilistic trees as discussed above. An observation is in the set corresponding to the fuzzy formula if the root node of the observation satisfies the formula interpreted as a traditional mu-calculus formula: so $\langle a \rangle \psi$ holds of an observation if the root has an a -transition leading to the root of an observation satisfying ψ , while it satisfies $[a]\psi$ if every a -transition (note, there can be at most one such a transition) leads to an observation that satisfies ψ . Conjunction and disjunction have their usual interpretation. $\mu X.\psi$ and $\nu X.\psi$ are fixpoint operators describing the “least” and “greatest” solutions, respectively, to the “equation” $X = \psi$. It will turn out that any state in a given RPLTS defines a probability space over observations and that our syntactic restrictions ensure that the sets of observations defined by any fuzzy formula are *measurable* in a precise sense³. State formulae will then be interpreted with respect to states in RPLTSs, with a state s satisfying a formula of the form $\text{Pr}_{\geq p}\psi$ if the measure of s -rooted observations satisfying ψ is at least p .

We will now consider some examples, before formally discussing the semantics (of the logic), to illustrate its power. In reading these formulae $\langle \cdot \rangle \phi$ is to be understood as the disjunction $\bigvee_{a \in Act} \langle a \rangle \phi$ and $[\cdot] \phi$ as $\bigwedge_{a \in Act} [a] \phi$.

- The formula $P_{\geq 1}(\nu X.\phi \wedge [.]X)$ says that it is almost always true that ϕ holds at all even time instants. It should be noted that the notion of “every even instance” can not be expressed in CTL*, and thus can not be expressed in PCTL*.
- The formulae $P_{\geq p}(\nu X.(P_{\leq q}\mu Y.[reset]Y) \wedge [step]X)$ expresses the requirement that measure of paths involving *steps* must be at least p , where at each *step* measure of finite-length reset paths is no more than q . In other words, we want probability of “infinite resets” at any stage to be sufficiently small, and the measure of these paths to be sufficiently big.
- Now consider the following formula used in reasoning about the blinkers of an automatic car [Nar99]. The formula

$$P_{\geq 1}[blinker](\mu X.(\langle reset \rangle true \vee ([blinker] false \wedge \langle \cdot \rangle X)))$$

means that it is almost always true that when the blinker is input then a reset occurs before the blinker is input again.

³ Measurability of formulae that permit alternation is still open.

Note that replacing $[blinker]$ by $\langle blinker \rangle$ would demand that the blinker be turned on, i.e., there be a transition to turn on the blinker in the current state.

- An example from the context of reasoning about a group membership protocol [Nar99] is the formula: $P_{\geq 1}(\langle tmrExp \rangle (noA \wedge noB))$ which essentially says that it is almost always true when the *timer* expires that neither A nor B is a member of the group.
- Finally, consider the following formula [Nar99], for some fixed p . The formula

$$P_{\geq p}\mu X.[(P_{\geq 1}\nu Y.[(incA \vee incB) \wedge \langle \cdot \rangle Y]) \vee \langle repA \rangle X \vee \langle repB \rangle X]$$

states that with probability p it is possible to do a finite number of $repA$ and $repB$ actions and enter a state from which only inconsistent states (either process A is in inconsistent state or process B is in inconsistent state) are descendants. That it, with probability p it is possible to get into an inconsistent state and remain inconsistent forever.

2.3 Semantics of GPL

In the remainder of this section we define the semantics of GPL formulae with respect to a fixed RPLTS $L = (S, \delta, P, I)$ by giving mutually recursive definitions of a relation $\models_L \subseteq S \times \Phi$ and a function $\Theta_L : \Psi \rightarrow 2^{\mathcal{M}_L}$. The former indicates when a state satisfies a state formula, while the latter returns the set of maximal d-trees satisfying a given fuzzy formula.

Our intention in defining $\Theta_L(\psi)$ is that it return trees which, when interpreted as (non-probabilistic) labeled transition systems, satisfy ψ interpreted as a mu-calculus formula. To this end, we augment Θ_L with an extra environment parameter $e : Var \rightarrow 2^{\mathcal{M}_L}$ that is used to interpret free variables. The formal definition of Θ_L is the following.

Definition 9 *The function Θ_L is defined inductively as follows.*

- $\Theta_L(\phi)e = \cup_{s \models_L \phi} \mathcal{M}_{L,s}$, where ϕ is a closed formula,
- $\Theta_L(X)e = e(X)$
- $\Theta_L(\langle a \rangle \psi)e = \{T \in \mathcal{M}_L \mid \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\psi)e\}$
- $\Theta_L([a]\psi)e = \{T \in \mathcal{M}_L \mid (T \xrightarrow{a} T') \Rightarrow T' \in \Theta_L(\psi)e\}$
- $\Theta_L(\psi_1 \wedge \psi_2)e = \Theta_L(\psi_1)e \cap \Theta_L(\psi_2)e$
- $\Theta_L(\psi_1 \vee \psi_2)e = \Theta_L(\psi_1)e \cup \Theta_L(\psi_2)e$
- $\Theta_L(\mu X.\psi)e = \cup_{i=0}^{\infty} M_i$, where $M_0 = \emptyset$ and $M_{i+1} = \Theta_L(\psi)e[X \mapsto M_i]$.
- $\Theta_L(\nu X.\psi)e = \cap_{i=0}^{\infty} N_i$, where $N_0 = \mathcal{M}_L$ and $N_{i+1} = \Theta_L(\psi)e[X \mapsto N_i]$.

When ψ has no free variables, $\Theta(\psi)e = \Theta(\psi)e'$ for any environments e, e' . In this case we drop the environment e and write $\Theta_L(\psi)$.

Some comments about this definition are in order. Firstly, it is straightforward to show that the semantics of all the operators except μ and ν are those that would be obtained by interpreting maximal deterministic trees as labeled transition systems and fuzzy formulae as mu-calculus formulae in the usual style [Koz83]. Secondly, because d-trees are deterministic it follows that if $T \in \Theta_L(\langle a \rangle \psi)$ then $T \in \Theta_L([a]\psi)$. Finally, the definitions we have given for μ and ν differ from the more general accounts that rely on the Tarski-Knaster fixpoint theorem. However, because of the “alternation-free” restriction we impose on our logic and the fact that d-trees are deterministic, the meanings of $\mu X.\psi$ and $\nu X.\psi$ are still least and greatest fixpoints in the usual sense.

We now remark on an important property of Θ_L . For a given $s \in S$ let $\Theta_{L,s}(\psi) = \Theta_L(\psi) \cap \mathcal{M}_L(s)$ be the maximal d-trees from s “satisfying” ψ . We then have the following.

Theorem 1 *For any $s \in S$ and $\psi \in \Psi$, $\Theta_{L,s}(\psi)$ is measurable.*

Proof The proof depends on the following observation.

Let ψ be a μ - and ν -free formula with free variable X , and let e be an environment. Then the function $f : 2^{\mathcal{M}_L} \rightarrow 2^{\mathcal{M}_L}$ given by $f(M) = \Theta(\psi)e[X \mapsto M]$ is continuous.

Clearly, the conjunction and disjunction operators are continuous. Consider the function

$$f_a(M) = \{T \in \mathcal{M}_L \mid T \xrightarrow{a} T', T' \in M\}$$

and a family of trees $\{M_i \subseteq \mathcal{M}_L\}_{i \geq 0}$, such that $\forall i \geq 0. M_i \subseteq M_{i+1}$. To show continuity of f_a , the operator corresponding to $\langle a \rangle$ modality, we argue as follows:

$$\begin{aligned} T \in \cup_{i \geq 0} f_a(M_i) &\text{ iff } \exists i. T \in f_a(M_i) \\ &\text{ iff } \exists i. \exists T' \in M_i. T \xrightarrow{a} T' \\ &\text{ iff } \exists T' \in \cup_{i \geq 0} M_i. T \xrightarrow{a} T' \\ &\text{ iff } T \in f_a(\cup_{i \geq 0} M_i) \end{aligned}$$

The argument for the $[a]$ modality is similar to the $\langle a \rangle$ modality because every node in a maximal tree has at most one outgoing edge for any action $a \in Act$. Consequently, non-fixpoint operators in the logic are continuous.

This continuity result implies that the semantics of μ and ν may be given iteratively; in particular, if ψ is μ - and ν -free with free variable X then we have that $\Theta(\mu X.\psi)e = \cup_{i=0}^{\infty} M_i$ and $\Theta(\nu X.\psi)e = \bigcap_{i=0}^{\infty} \widehat{M}_i$, where $M_0 = \emptyset$ and $M_{i+1} = \Theta(\psi)e[X \mapsto M_i]$, and $\widehat{M}_0 = \mathcal{M}_L$ and $\widehat{M}_{i+1} = \Theta(\psi)e[X \mapsto \widehat{M}_i]$.

It is easy to show that each M_i is measurable, and as countable unions of measurable sets are also measurable, the result follows (and dually for \widehat{M}_i). \square

We can now define the semantics of state formulae by defining the relation \models_L .

Definition 10 *Let $L = (S, \delta, P, I)$ be a RPLTS. Then \models_L is defined inductively as follows.*

- $s \models_L A$ iff $A \in I(s)$.
- $s \models_L \neg A$ iff $A \notin I(s)$.
- $s \models_L \phi_1 \wedge \phi_2$ iff $s \models_L \phi_1$ and $s \models_L \phi_2$.
- $s \models_L \phi_1 \vee \phi_2$ iff $s \models_L \phi_1$ or $s \models_L \phi_2$.
- $s \models_L \text{Pr}_{>p}\psi$ iff $\mathbf{m}_s(\Theta_{L,s}(\psi)) > p$.
- $s \models_L \text{Pr}_{\geq p}\psi$ iff $\mathbf{m}_s(\Theta_{L,s}(\psi)) \geq p$.

An atomic proposition is satisfied by a state if the proposition is a member of the propositional labeling of the state. Conjunction and disjunction are interpreted in the usual manner, while a state satisfies a formula $\text{Pr}_{>p}\psi$ iff the measure of the observations of ψ rooted at s exceeds p , and similarly for $\text{Pr}_{\geq p}\psi$.

We now establish some important properties of GPL. The first shows that the modal operators for fuzzy formulae enjoy certain distributivity laws with respect to the propositional operators.

Lemma 1 *For a RPLTS L , fuzzy formulae ψ_1 and ψ_2 and $a \in \text{Act}$, we have:*

- (1) $\Theta_L(\langle a \rangle(\psi_1 \vee \psi_2)) = \Theta_L(\langle a \rangle\psi_1 \vee \langle a \rangle\psi_2)$
- (2) $\Theta_L([a](\psi_1 \vee \psi_2)) = \Theta_L([a]\psi_1 \vee [a]\psi_2)$
- (3) $\Theta_L(\langle a \rangle(\psi_1 \wedge \psi_2)) = \Theta_L(\langle a \rangle\psi_1 \wedge \langle a \rangle\psi_2)$
- (4) $\Theta_L([a](\psi_1 \wedge \psi_2)) = \Theta_L([a]\psi_1 \wedge [a]\psi_2)$
- (5) $\Theta_L([a]\psi_1 \wedge \langle a \rangle\psi_2) = \Theta_L(\langle a \rangle(\psi_1 \wedge \psi_2))$

Proof Most cases are straightforward and are omitted. That $\langle a \rangle$ distributes over \wedge and $[a]$ over \vee is due to the determinacy of d-trees. We prove the former

of these statements; the latter is similar.

$$\begin{aligned}
& \Theta_L(\langle a \rangle(\psi_1 \wedge \psi_2))e \\
&= \{T | \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\psi_1 \wedge \psi_2)e\} \\
&= \{T | \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\psi_1)e \cap \Theta_L(\psi_2)e\} \\
&= \{T | \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\psi_1)e\} \cap \{T | \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\psi_2)e\} \\
&\hspace{15em} (\text{follows from determinacy of } T) \\
&= \Theta_L(\langle a \rangle\psi_1 \wedge \langle a \rangle\psi_2)
\end{aligned}$$

□

Based on Theorem 1 and the definition of Θ_L , the next lemma also holds.

Lemma 2 *Let $s \in S$, $a \in \text{Act}$ and $\psi, \psi_1, \psi_2 \in \Psi$. Then we have the following.*

$$m_s(\Theta_{L,s}(\psi_1 \vee \psi_2)) = m_s(\Theta_{L,s}(\psi_1)) + m_s(\Theta_{L,s}(\psi_2)) - m_s(\Theta_{L,s}(\psi_1 \wedge \psi_2)) \quad (1)$$

$$m_s(\Theta_{L,s}(\langle a \rangle\psi)) = \sum_{s':(s,a,s') \in \delta} P(s, a, s') * m_{s'}(\Theta_{L,s'}(\psi)) \quad (2)$$

$$m_s(\Theta_{L,s}([a]\psi)) = \begin{cases} m_s(\Theta_{L,s}(\langle a \rangle\psi)) & \text{if } (s, a, s') \in \delta \text{ for some } s' \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

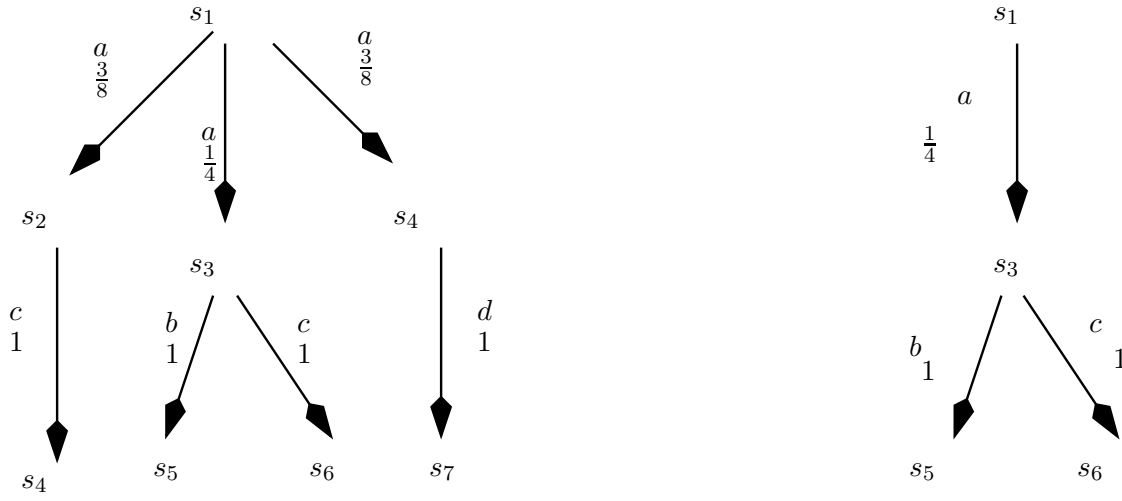
Proof The first statement follows from standard properties of probability spaces. We will prove the second statement; the proof of the last statement is similar. Now

$$\Theta_{L,s}(\langle a \rangle\psi) = \bigcup_{s':s \xrightarrow{a} s'} \{T \in \mathcal{M}_L(s) | \exists T' \in \Theta_{L,s'}(\psi). T \xrightarrow{a} T'\}$$

Note that if $s' \neq s''$ then $\Theta_{L,s'}(\psi) \cap \Theta_{L,s''}(\psi) = \emptyset$. Therefore, it follows from the inclusion-exclusion principle of probability spaces⁴ that $m_s(\Theta_L(\langle a \rangle\psi)) = \sum_{(s,a,s') \in \delta} P(s, a, s') * m_{s'}(\Theta_L(\psi))$. □

Finally, although our logic only allows a restricted form of negation, we do have the following.

⁴ A standard condition that a probability function should satisfy is $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$, which is an extension of Inclusion-Exclusion principle.



Tree T

Let $I(s_5) = I(s_6) = I(s_7) = I(s_8) = \{\sigma\}$

Fig. 2. A small example

Lemma 3 *Let $L = (S, \delta, P, I)$ be a RPLTS, let $s \in S$, and let ψ and ϕ be fuzzy and state formulae, respectively. Then there exist formulae $\text{neg}(\psi)$ and $\text{neg}(\phi)$ such that:*

$$\Theta_{L,s}(\text{neg}(\psi)) = \mathcal{M}_{L,s} - \Theta_{L,s}(\psi) \quad \text{and} \quad s \models_L \text{neg}(\phi) \Leftrightarrow s \not\models_L \phi.$$

Proof We can prove this conjecture by induction on the structure of the formula, where the induction hypothesis includes both kinds of formulae. In the process we would use the duality of \wedge and \vee , duality of $[a]$ and $\langle a \rangle$, duality of ν and μ , and the duality of $\text{Pr}_{>p}$ and $\text{Pr}_{\geq 1-p}$. \square

2.4 Example Specifications

We will consider a couple of examples in this section to illustrate our specification language.

Example 1 Consider the acyclic RPLTS on the left hand side of Figure 2. Suppose $\sigma \in \text{Prop}$ such that $\forall s \in \{s_5, s_6, s_7, s_8\} : \sigma \in I(s)$. Now consider the formulae:

$$\psi_1 = \langle a \rangle \langle b \rangle \sigma \wedge \langle a \rangle \langle c \rangle \sigma$$

$$\psi_2 = \langle a \rangle \langle c \rangle \sigma \wedge \langle a \rangle \langle d \rangle \sigma$$

Note that $\Theta_{L,s_1}(\psi_2) = \emptyset$ as none of the d-trees of L rooted at s_1 satisfy the requirements of ψ_2 . However, $\Theta_L(\psi_1) = \{T\}$, as shown in Figure 2. Given that

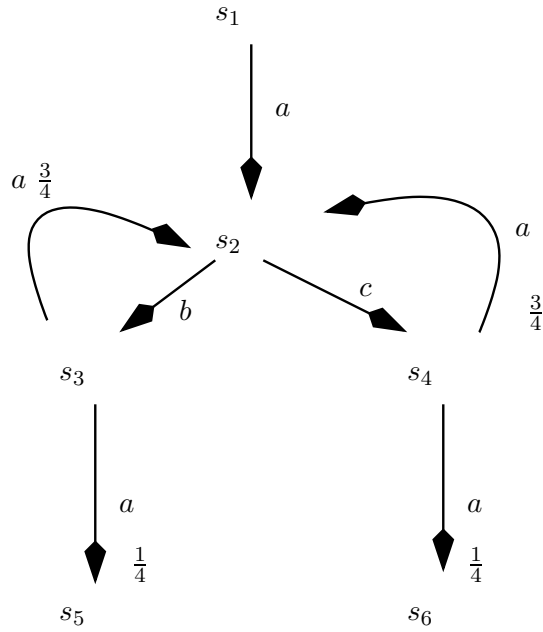


Fig. 3. An example specification with loops

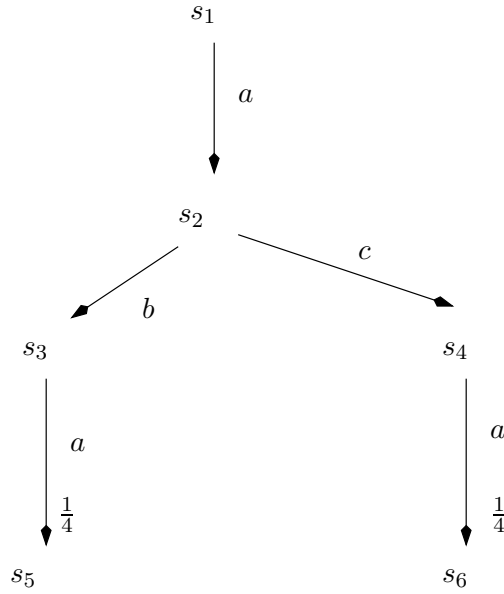


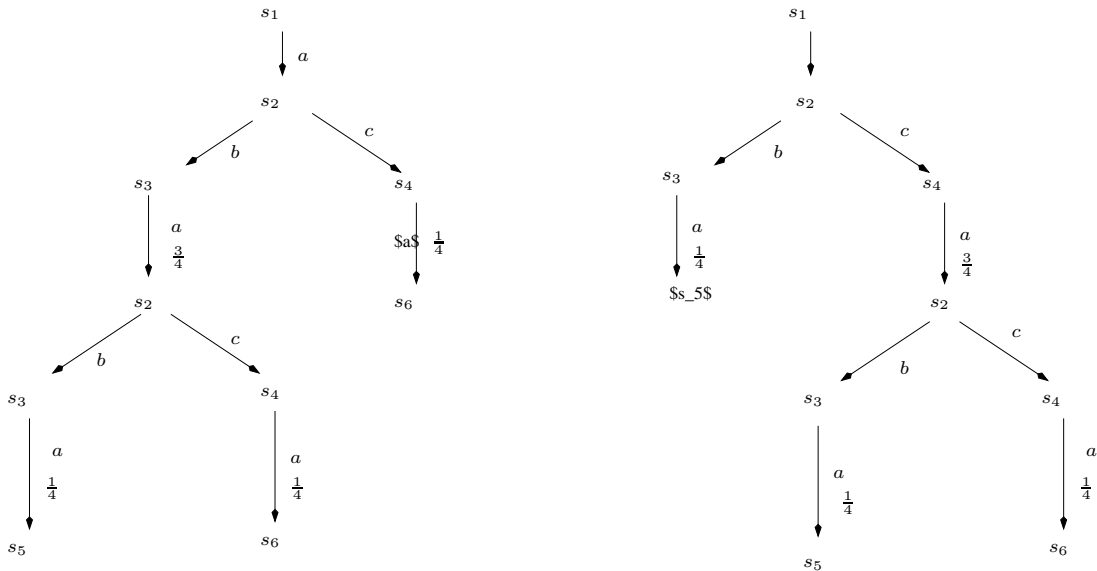
Fig. 4. An observation tree for specification from Figure 3

T is maximal and $\mathbf{m}_{s_1}(\{T\}) = 1/4$ we conclude that $s \models \text{Pr}_{>p}(\psi_1 \vee \psi_2)$ holds for any $p < 1/4$. Further, we have $\Theta_L(\psi_1 \wedge \psi_2) = \emptyset$ and $\mathbf{m}_{s_1}(\Theta_{L,s_1}(\psi_1 \wedge \psi_2)) = 0$.

Now let us consider an example RPLTS that has loops in its specification.

Example 2 Consider the RPLTS given in Figure 3, and the fuzzy formula

$$\psi_3 = \mu X.([a][b]X \wedge [a][c]X)$$



Note that s_2 is visited several times

Fig. 5. Two observation trees for specification in Figure3

Since we have a mu-formula, observations that satisfy the formula will have paths that leave the loop from state s_2 . Furthermore, given the semantics of $[b]X$ and $[c]X$, all finite observation trees rooted at s_1 satisfy the mu-formula. The first few observation trees, among unboundedly many, are presented in figures 4 and 5. Thus, it is easy to see that ψ_3 characterizes in the system presented in Figure 3 those observations that are finite and, thus, end in states s_5 or s_6 on every path. To calculate the probability of all these finite height observations, consider the following facts:

- Nodes labeled s_2 are the only ones with two children and all other nodes have a single child.
- Except for the occurrence of s_2 near the root, all other occurrences of s_2 have an a transition entering them with probability $\frac{3}{4}$.
- Edges entering leaves are a transitions with probability $\frac{1}{4}$.
- All observations can be looked upon as binary trees (by short-circuiting all nodes with outdegree 1).

Now using the fact that the number of internal nodes in a binary tree is one less than the number of leaves, the probability of all the observations is

$$p = \sum_{n \geq 1} C_n \cdot \left(\frac{3}{4}\right)^{n-1} \cdot \left(\frac{1}{4}\right)^{n+1}$$

where C_n is the number of binary trees with n internal nodes. Note that a tree with n internal nodes (i.e., labeled by s_2) has $(n - 1)$ a -transitions each with probability $\frac{3}{4}$ (not counting the $s_1 \xrightarrow{a} s_2$ transition) and $(n + 1)$ a -transitions each with probability $\frac{1}{4}$. It turns out the number of trees C_n is the n^{th} Catalan

number given by the formula $\frac{(2n)!}{n!(n+1)!}$. Furthermore, the generating sequence for Catalan numbers is⁵

$$g(x) = \frac{1 - \sqrt{1 - 4x}}{2x} = \sum_{n \geq 0} C_n x^n = 1 + C_1 x + C_2 x^2 + \dots$$

Using this sequence we can calculate the required probability p as:

$$\begin{aligned} p &= \sum_{n \geq 1} C_n \left(\frac{3}{4}\right)^{n-1} \left(\frac{1}{4}\right)^{n+1} = \frac{1}{3} \cdot \sum_{n \geq 1} C_n \left(\frac{3}{16}\right)^n = \frac{1}{3} \cdot \left(g\left(\frac{3}{16}\right) - 1\right) = \\ &= \frac{1}{3} \cdot \left(\frac{4}{3} - 1\right) = \frac{1}{9} \end{aligned}$$

Thus, we have $\mathbf{m}_{s_1}(\Theta_{L,s_1}(\psi_3)) = \frac{1}{9}$.

We now consider variations of the formula ψ_3 in the next example.

Example 3 Consider the system given in Figure 3. Furthermore, consider the formula

$$\psi_4 = \nu X.([a][b]X \wedge [a][c]X)$$

Since all maximal observations satisfy ψ_4 we have $\mathbf{ms}_1(\Theta_{L,s_1}(\psi_4)) = 1$.

3 Expressiveness of GPL

In this section we illustrate the expressive power of GPL by showing how two different probabilistic logics may be encoded within our logic. We also discuss how our approach differs from that of Huth and Kwiatkowska [HK97] who also interpret Mu-calculus against Reactive systems.

3.1 Encoding Probabilistic Modal Logic

Probabilistic Modal Logic (PML) [LS91] is a probabilistic version of Hennessy-Milner logic [HM85] that has been shown to characterize probabilistic bisimulation equivalence over RPLTSs. The formulae of the logic are generated by the following grammar:

$$\phi ::= A \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \langle a \rangle_p \phi$$

⁵ This information is available on the web at the *Math-World* website: mathworld.wolfram.com/BinaryTree.html and mathworld.wolfram.com/CatalanNumber.html!

where $0 \leq p < 1$, $A \in Prop$ and $a \in Act$. Note that the definition of PML [LS91] takes $Prop = \{\mathbf{tt}, \mathbf{ff}\} \cup \{\Delta_a \mid a \in Act\}$ where Δ_a is true of states from which there are no a labeled transitions. Formulae are interpreted with respect to states in a given RPLTS $L = (S, \delta, P, I)$ via a relation $\models_L^{PML} \subseteq S \times \phi$. The definition appears below; the cases for \neg and \wedge have been omitted.

$$s \models_L^{PML} A \quad \text{iff } A \in I(s)$$

$$s \models_L^{PML} \langle a \rangle_p \phi \quad \text{iff } \sum_{\{s' \mid (s,a,s') \in \delta \wedge s' \models_L^{PML} \phi\}} P(s, a, s') \geq p$$

Note that a state s satisfies $\langle a \rangle_p \phi$ provided that the probability of taking an a -transition to a state satisfying ϕ is at least p . This observation suggests the following encoding function E_{PML} for translating PML formulae into GPL formulae.

$$E_{PML}(\phi) = \begin{cases} \phi & \text{if } \phi \in Prop \\ E_{PML}(\phi_1) \wedge E_{PML}(\phi_2) & \text{if } \phi = \phi_1 \wedge \phi_2 \\ \text{neg}(E_{PML}(\phi')) & \text{if } \phi = \neg\phi' \\ \text{Pr}_{\geq p} \langle a \rangle E_{PML}(\phi') & \text{if } \phi = \langle a \rangle_p \phi' \end{cases}$$

In essence, the translation effectively replaces all occurrences of $\langle a \rangle_p$ by $\text{Pr}_{\geq p} \langle a \rangle$. We have the following:

Theorem 2 *Let ϕ be a PML formula and s be a state of RPLTS L . Then $s \models_L^{PML} \phi$ iff $s \models E_{PML}(\phi)$.*

Proof By induction on the structure of the PML formulae and the following observation: if ϕ' is a state formula of GPL then

$$\begin{aligned} & \mathbf{m}_s(\{T \mid \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\phi')\}) \\ &= \mathbf{m}_s(\{T \mid \exists T' : T \xrightarrow{a} T' \wedge \text{root}(T') \models \phi'\}) \\ &= \sum_{s'.(s,a,s') \in \delta \wedge s' \models \phi'} P(s, a, s') \end{aligned}$$

□

3.2 Encoding pCTL*

pCTL* [ASB⁺95] is a probabilistic variant of the temporal logic CTL* [EH86]. The latter logic is interpreted with respect to Kripke structures; the former

is interpreted with respect to *Markov processes* (MP) [ASB⁺95], which are discrete-time Markov chains; these may be viewed as probabilistic Kripke structures. It turns out that MPs form a subclass of RPLTSs. This section will show that pCTL* has an uniform encoding in GPL.

A Markov process may be seen as a RPLTS having only one action label and in which every state has at least one outgoing transition.

Definition 11 *Let $Act = \{a\}$. Then a Markov process (MP) is a RPLTS (S, δ, P, I) such that for any $s \in S$, $\sum_{\{s' \mid (s,a,s') \in \delta\}} P(s, a, s') = 1$.*

It is straightforward to see that the d-trees of a MP are in fact isomorphic to sequences of states from the MP: a sequence $\pi = s_0 s_1 \dots$ coincides with the d-tree $\{\sigma_0, \sigma_1, \dots\}$, where $\sigma_0 = s_0$ and $\sigma_{i+1} = \sigma_i \xrightarrow{a} s_{i+1}$. It then turns out that the measure space of d-trees for a state in a MP coincides with the standard sequence space construction for Markov chains [KSK66]. Consequently, in the following we will use the function m_s to refer to the measure of both sets of sequences and sets of d-trees. We also use the following notations on infinite sequences $\pi = s_0 s_1 \dots$, $\pi[i] = s_i$, and $\pi^i = s_i s_{i+1} \dots$.

3.2.1 Interpreting GPL over Markov chains

As every state in a MP has an outgoing transition, the semantics of the GPL constructs $\langle a \rangle \psi$ and $[a] \psi$ coincide. That is, when L is a MP it follows from Definition 9 that $\Theta_L(\langle a \rangle \psi) = \Theta_L([a] \psi)$.

3.2.2 pCTL*

Let *Prop* be a set of atomic propositions. The grammar below summarizes the syntax of pCTL*, which has two levels—state formulae (ϕ) and *path* formulae (ψ). State formulae specify properties that hold in states of a MP while path formulae specify properties of execution sequences.

$$\begin{aligned} \phi &::= A \mid \neg \phi \mid \phi_1 \vee \phi_2 \mid Pr_{<p} \psi \mid Pr_{>p} \psi \\ \psi &::= \phi \mid \neg \psi \mid \psi_1 \vee \psi_2 \mid X \psi \mid \psi_1 \mathbf{U} \psi_2 \end{aligned}$$

Here $Pr_{>p}$ and $Pr_{<p}$ are probabilistic quantifiers, while X denotes the next-state and \mathbf{U} the until operator, respectively.

The semantics of pCTL* formulae is given with respect to a MP $L = (S, \delta, P, I)$ via a relation \models_L relating states in L to state formulae and paths (infinite state sequences) in L to path formulae. The interpretations of \neg and \vee are standard,

and we omit them; what follows defines the meanings of the other operators.

$$\begin{aligned}
s &\models_L^{pCTL^*} A && \text{iff } A \in I(s) \\
s &\models_L^{pCTL^*} Pr_{<p}\psi && \text{iff } \mathfrak{m}_s(\{\pi \mid \pi \models_L^{pCTL^*} \psi\}) < p \\
s &\models_L^{pCTL^*} Pr_{>p}\psi && \text{iff } \mathfrak{m}_s(\{\pi \mid \pi \models_L^{pCTL^*} \psi\}) > p \\
\pi &\models_L^{pCTL^*} \phi && \text{iff } \pi[0] \models_L^{pCTL^*} \phi \\
\pi &\models_L^{pCTL^*} X\psi && \text{iff } \pi^1 \models_L^{pCTL^*} \psi \\
\pi &\models_L^{pCTL^*} \psi_1 \mathbf{U} \psi_2 && \text{iff } \exists k \geq 0 : \pi^k \models_L^{pCTL^*} \psi_2 \wedge \forall j : 0 \leq j < k : \pi^j \models_L^{pCTL^*} \psi_1
\end{aligned}$$

Our encoding of pCTL* in GPL translates state formulae into state formulae and path formulae into fuzzy ones. Our approach relies on the following recursive characterization of \mathbf{U} : $\pi \models_L \psi_1 \mathbf{U} \psi_2$ iff $\pi \models_L \psi_2 \vee (\psi_1 \wedge X(\psi_1 \mathbf{U} \psi_2))$. The encoding may now be given as a function E_{pCTL^*} as follows, where γ is either a state formula or a path formula.

$$E_{pCTL^*}(\gamma) = \begin{cases} \gamma & \text{if } \gamma \in Prop \\ \text{neg}(E_{pCTL^*}(\gamma')) & \text{if } \gamma = \neg\gamma' \\ E_{pCTL^*}(\gamma_1) \vee E_{pCTL^*}(\gamma_2) & \text{if } \gamma = \gamma_1 \vee \gamma_2 \\ Pr_{\geq 1-p}\text{neg}(E_{pCTL^*}(\psi)) & \text{if } \gamma = Pr_{<p}\psi \\ Pr_{>p}E_{pCTL^*}(\psi) & \text{if } \gamma = Pr_{>p}\psi \\ \langle a \rangle E_{pCTL^*}(\psi) & \text{if } \gamma = X\psi \\ \mu X.(E_{pCTL^*}(\psi_2) \vee (E_{pCTL^*}(\psi_1) \wedge \langle a \rangle X)) & \text{if } \gamma = \psi_1 \mathbf{U} \psi_2 \end{cases}$$

The translation given above might come as a surprise to some readers as it is well known that CTL* can not be encoded in alternation-free mu-calculus. The reason why our encoding succeeds is because the path formula ψ , in the context of $Pr_{>r}\psi$, is only interpreted over paths (not trees) and thus treated as an LTL formula which can be easily expressed in the alternation-free μ -calculus [Sti92]. We now have the following:

Theorem 3 *Let L be a MP, let s be a state in L , and let π be a path in L . Then the following holds:*

- (1) *For any pCTL* state formula ϕ , $s \models_L^{pCTL^*} \phi$ iff $s \models_L E_{pCTL^*}(\phi)$.*
- (2) *For any pCTL* path formula ψ , $\pi \models_L^{pCTL^*} \psi$ iff $\pi \in \Theta_L(E_{pCTL^*}(\psi))$*

Proof Follows from the fact that d-trees of MPs are sequences and the fact

that the LTL formula $\psi_1 \mathbf{U} \psi_2$ can be encoded in the linear-time mu-calculus as $\mu X.(\psi_2 \vee (\psi_1 \wedge \langle a \rangle X))$. \square

3.3 Reconstructing the Logic of Huth and Kwiatkowska

Huth and Kwiatkowska develop a notion of *quantitative model checking* [HK97] in which one calculates the likelihood with which a system state satisfies a formula. The basis for their approach lies in a semantics for the modal mu-calculus that assigns “probabilities”, rather than truth values, to assertions about states in a RPLTS. In this section we briefly review their approach, offer a criticism of it, and show how GPL provides a principled means of remedying the criticism.

The syntax of their logic coincides with the syntax of our fuzzy formulae with the following exceptions: (1) they allow negation (although in such a way that negations can be eliminated in the usual manner); (2) the only atomic propositions are **tt** (“true”) and **ff** (“false”); (3) no use of the probabilistic quantifiers $\mathbf{Pr}_{\geq p}$ and $\mathbf{Pr}_{> p}$ is allowed. They then present three semantics for the logic that differ only in their interpretation of conjunction. Each interprets formulae as functions mapping states to numbers in $[0, 1]$; formally, given RPLTS L , $\llbracket \psi \rrbracket_L : S \rightarrow [0, 1]$ represents the interpretation of formula ψ . What follows presents the relevant portions of these semantics.

$$\begin{aligned} \llbracket \mathbf{tt} \rrbracket_L(s) &= 1 \\ \llbracket \langle a \rangle \psi \rrbracket_L(s) &= \sum_{s':(s,a,s') \in \delta} P(s, a, s') \cdot \llbracket \psi \rrbracket_L(s') \\ \llbracket \psi_1 \wedge \psi_2 \rrbracket_L(s) &= f(\llbracket \psi_1 \rrbracket_L(s), \llbracket \psi_2 \rrbracket_L(s)) \end{aligned}$$

The meanings of the other boolean and modal operators may be obtained using dualities (e.g. $\llbracket [a]\psi \rrbracket_L(s) = 1 - (\llbracket \langle a \rangle \neg \psi \rrbracket_L(s))$), while the meanings of fixed points may be obtained using the usual Tarski-Knaster construction. The semantics of \wedge contains a parameter f ; [HK97] provides three different instantiations of f .

- (1) $f(x, y) = \min(x, y)$
- (2) $f(x, y) = x \cdot y$
- (3) $f(x, y) = \max(x + y - 1, 0)$

Each unfortunately has its drawbacks. The first two fail to validate some expected logical equivalences; for example it is not the case that **tt** is equivalent to $\psi \vee \neg \psi$. The authors refer to the third as a “fuzzy” interpretation and indicate that it is intended only to provide a “lower approximation” on probabilities.

GPL permits a similar interpretation to be attached to the mu-calculus, but in such a way that a unique probability value is attached to any formula. Consider the function $\llbracket \psi \rrbracket_L^{GPL}$ given by:

$$\llbracket \psi \rrbracket_L^{GPL}(s) = \mathbf{m}_s(\Theta_L(\psi)).$$

One can show that this interpretation preserves much of the semantics of Huth and Kwiatkowska; in particular, Lemmas 2 and 3 show that this definition attaches the same interpretations to the modalities. More importantly, the formulae $\psi \wedge \neg\psi$ and \mathbf{tt} have the same probability associated with them.

4 Model Checking

This section describes a procedure for determining whether or not a given state in a finite-state RPLTS satisfies a GPL formula. We present the algorithm in two stages. The first shows how to calculate the measure of observations that are rooted at a given RPLTS state and satisfy a fuzzy formula; the second then shows how this routine may be used to implement full GPL model checking. We assume that the formulae to be considered have no *unguarded* occurrences of bound variables. That is, in every subformula of the form $\sigma X.\psi$, where σ is either μ or ν , each occurrence of X in ψ falls within the scope of a $\langle a \rangle$ or a $[a]$ operator. Any mu-calculus formula may be transformed into one satisfying this restriction [KVV00]. In the remainder of this section we fix a specific RPLTS $L = (S, \delta, P, I)$.

4.1 Computing the Measure of Fuzzy Formulae

We will now describe a procedure *modchk-fuzzy* whose task is to compute $\mathbf{m}_{s_0}(\Theta_{L,s_0}(\psi))$ for a given fuzzy formula ψ and a state s_0 of the RPLTS. The algorithm consists of the following steps.

- (1) From L , s_0 and ψ , construct a dependency graph.
- (2) From the graph, extract a system of (non-linear) *measure* equations.
- (3) Calculate a specific solution to these equations which will be the $\mathbf{m}_{s_0}(\Theta_{L,s_0}(\psi))$.

We now describe each of the three steps and prove its correctness.

4.1.1 A Graph Construction

Fix a state s_0 and a fuzzy formula ψ . The first step in *modchk-fuzzy* involves constructing a graph $\text{Pr}(s_0, \psi)$ that describes the relationship between the quantity $\mathbf{m}_{s_0}(\Theta_{L,s_0}(\psi))$, that we wish to compute, and quantities of the form $\mathbf{m}_s(\Theta_{L,s}(\psi'))$, where s is a state reachable from s_0 and ψ' is an (appropriate) subformula of ψ . This graph will have vertices of the form (s, F) , where $s \in S$ and F is a set of fuzzy formulae, with an associated semantics of

$$\llbracket (s, F) \rrbracket = \Theta_{L,s}(\wedge F)$$

The edges from (s, F) then provide “local” information regarding $\llbracket (s, F) \rrbracket$.

In order to define the graph formally we need the following notions.

Definition 12 For a closed fuzzy formula ψ define the (Fisher-Ladner) closure, written as $Cl(\psi)$, as the smallest set of formulae satisfying the following rules:

- $\psi \in Cl(\psi)$
- 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 $\psi''[\sigma X. \psi'' / X] \in Cl(\psi)$ (σ is either μ or ν)

One may easily show that $Cl(\psi)$ contains no more elements than ψ contains subformulae.

The node set N in the graph is a subset of the set $S \times 2^{Cl(\psi)}$; that is, nodes have form (s, F) , where $s \in S$ and $F \subseteq Cl(\psi)$. A node is classified according to the first rule, among the following, that it satisfies.

- (s, F) is an *empty node* if $F = \emptyset$.
- (s, F) is a *false node* if there exists a state formula $\phi \in F$ with $s \not\models_L \phi$ or if there exists a formula of the form $\langle a \rangle \psi'$ and there is no s' such that $s \xrightarrow{a} s'$.
- (s, F) is a *true node* if (a) it has at least one state formula $\phi \in F$ such that $s \models \phi$ or (b) it has at least one $[a] \psi \in F$ but there are no transitions of the form $s \xrightarrow{a} s'$ for any $s' \in S$.
- (s, F) is an *and-node* if there exists a formula $\psi_1 \wedge \psi_2 \in F$.
- (s, F) is a ν -node if F has a formula of the form $\nu X. \psi'$.
- (s, F) is a μ -node if F has a formula of the form $\mu X. \psi'$.
- (s, F) is an *or-node* if there exists a formula $\psi_1 \vee \psi_2 \in F$.
- (s, F) is an *action-node* if every formula in F has form $\langle a \rangle \psi'$ or $[a] \psi'$.

As an example, a node $(s, \{[a] \psi', \mu X. \psi''\})$ will be classified as a true node if there is no s' such that $s \xrightarrow{a} s'$. It will, however, be considered a μ -node if

there is an s' such that $s \xrightarrow{a} s'$. In the following we will also assume that there is a linear order on the formulae in $Cl(\psi)$; this is easily achieved as $Cl(\psi)$ is a finite set. However, there are more sophisticated linear orderings which respect the subformula relation modulo unfoldings of recursive calls [Cle90].

Before we start describing the graph construction we need to define the function $\text{residue} : 2^\Psi \times Act \rightarrow 2^\Psi$ which removes modal operators from a set F of formulae.

$$\text{residue}(F, a) = \{\psi' \mid \langle a \rangle \psi' \in F \vee [a] \psi' \in F\}$$

The edges in the graph are labeled by elements drawn from the set $Act \cup \{\epsilon^+, \epsilon^-\}$ (where it is assumed that $\epsilon^+, \epsilon^- \notin Act$). The edge set $E \subseteq N \times (Act \cup \{\epsilon^+, \epsilon^-\}) \times N$ is defined as follows.

- (1) If $n = (s, F)$ is an empty node or a false node,⁶ then n is a terminal node;
- (2) else if (s, F) is a true node then $((s, F), \epsilon^+, (s, F')) \in E$, where F' is F with all state formulae and appropriate $[a]\psi'$ formulae deleted. Thus,

$$F' = F - \{\phi \in F \mid s \models \phi\} - \{[a]\psi \mid \neg \exists s'. s \xrightarrow{a} s'\}$$

- (3) else if (s, F) is a ν -node with the fixpoint formula of greatest rank (according to the linear order) being $\psi' = \nu X. \psi''$ then $((s, F), \epsilon^+, (s, F - \{\psi'\} \cup \{\psi''[\psi'/X]\})) \in E$;
- (4) else if (s, F) is a μ -node with the fixpoint formula of greatest rank being $\psi' = \mu X. \psi''$ then $((s, F), \epsilon^+, (s, F - \{\psi'\} \cup \{\psi''[\psi'/X]\})) \in E$;
- (5) else if (s, F) is an and-node with the conjunction formula of greatest rank being $\psi = \psi_1 \wedge \psi_2$ then $((s, F), \epsilon^+, (s, F - \{\psi\} \cup \{\psi_1, \psi_2\})) \in E$
- (6) else if (s, F) is an or-node with the disjunction formula of greatest rank being $\psi = \psi_1 \vee \psi_2 \in F$ then $((s, F), \epsilon^+, (s, F - \{\psi\} \cup \{\psi_1\})) \in E$, $((s, F), \epsilon^+, (s, F - \{\psi\} \cup \{\psi_2\})) \in E$, and $((s, F), \epsilon^-, (s, F - \{\psi\} \cup \{\psi_1, \psi_2\})) \in E$;
- (7) else if (s, F) is an action node, then for any $a \in Act$ with $\text{residue}(F, a) \neq \emptyset$ and $s' \in S$ such that $(s, a, s') \in \delta$, $((s, F), a, (s', \text{residue}(F, a))) \in E$.

The graph construction is guaranteed to terminate; this is due to the fact that all the formulae arising in the construction are in the Fisher-Ladner closure of ψ [Koz83]. Formally, we have:

Lemma 4 *For a finite-state RPLTS a state $s_0 \in S$ and a formulae ψ the number of nodes in $\text{Pr}(s_0, \psi)$ is bounded by $|S| \times 2^{|\psi|}$.*

⁶ Determining whether a node is false may require determining if $s \models_L \phi$ for some state formula. This can be done by (recursively) invoking the model-checking procedure described in the next section.

Turning to semantic issues, it should be noted that edges indicate a “local relationship” between $\llbracket (s, F) \rrbracket$ and $\llbracket (s', F') \rrbracket$ when (s, F) and (s', F') are related by an edge. More specifically, first note that if (s, F) is an empty node (a false node) then $\mathbf{m}_s(\Theta_{L,s}(\wedge F)) = 1(0)$. Now suppose that (s, F) has its children generated by Rule (6) above (i.e. is an or-node). This means that $\exists F'. F = F' \cup \{\psi_1 \vee \psi_2\} \wedge F \neq F'$, and the semantics of the logic entails that $\wedge F$ and $(\wedge F' \wedge \psi_1) \vee (\wedge F' \wedge \psi_2)$ are logically equivalent. From Lemma 2, Equation 1, we may therefore conclude the following.

$$\mathbf{m}_s(\Theta_{L,s}(\wedge F)) = \mathbf{m}_s(\Theta_{L,s}(\wedge F' \wedge \psi_1)) + \mathbf{m}_s(\Theta_{L,s}(\wedge F' \wedge \psi_2)) - \mathbf{m}_s(\Theta_{L,s}(\wedge (F' \cup \{\psi_1, \psi_2\})))$$

This observation is encoded in the ϵ^+ and ϵ^- edges emanating from (s, F) . Similar observations hold for the other nodes, the exception being of action nodes, whose treatment we discuss later.

To justify our graph construction consider the following operation on sets of trees $\mathbf{f}_{(s,a)} : 2^{\mathcal{M}_L} \rightarrow 2^{\mathcal{M}_L(s)}$ defined as:

$$\mathbf{f}_{(s,a)}(M) = \{T \in \mathcal{M}_L(s) \mid \exists T' \in M. T \xrightarrow{a} T'\}$$

In essence, the operator $\mathbf{f}_{(s,a)}$ extends trees by adding an edge of the form $s \xrightarrow{a} s'$ and “completing” the result into a maximal d-tree. We will use the following properties of this operator.

Lemma 5 *For any family of sets of maximal d-trees $\{M_i\}_{i \in \mathcal{I}}$ over an index set \mathcal{I} , we have the following:*

$$\bigcup_{i \in \mathcal{I}} \mathbf{f}_{(s,a)}(M_i) = \mathbf{f}_{(s,a)}\left(\bigcup_{i \in \mathcal{I}} M_i\right) \quad (4)$$

$$\bigcap_{i \in \mathcal{I}} \mathbf{f}_{(s,a)}(M_i) = \mathbf{f}_{(s,a)}\left(\bigcap_{i \in \mathcal{I}} M_i\right) \quad (5)$$

Both statements are obvious as there must be a maximal tree T in some M_i (for some $i \in \mathcal{I}$) which is extended in the first case, and there is a maximal d-tree T in all $M_i, i \in \mathcal{I}$, which is extended in the second case. Given that the semantics of formulae are full d-trees we make the following claim regarding the interaction between the $\mathbf{residue}_{(-,a)}$ and $\mathbf{f}_{(-,a)}$ operations, which follows from Lemma 5, Equation 5 and from the semantics of conjunction of formulae:

Lemma 6 *Let F be a set of modal formulae. Then for all $a \in \text{Act}$ and $s \in S$, we have:*

$$\bigcap_{\psi \in \mathbf{residue}(F,a)} \mathbf{f}_{(s,a)}(\Theta_L(\psi)) = \mathbf{f}_{(s,a)}\left(\Theta_L\left(\bigwedge_{\psi \in \mathbf{residue}(F,a)} \psi\right)\right)$$

A final operator we need returns the set of action symbols that appear as root

modal operators. To wit, we define $\text{action} : 2^\Psi \rightarrow 2^{\text{Act}}$ as

$$\text{action}(F) = \{a \in \text{Act} \mid \exists \psi. \langle a \rangle \psi \in F \vee [a] \psi \in F\}$$

Now consider an action node (s, F) . Note that the rule for an action node is applied if and only if (a) all formulae in F are of the form $\langle a \rangle \psi'$ or $[a] \psi'$ and (b) for every $\langle a \rangle \psi'$ or $[a] \psi'$ there is at least one a transition (s, a, s') for some s' . Now observe that a s -rooted d-tree T satisfies all of the formulae in F if and only if for every $a \in \text{action}(F)$, there is a tree T' such that $T \xrightarrow{a} T'$ and T' satisfies ψ' . We thus have the following characterization of the semantics of the nodes of the graph.

Lemma 7 *Fix a node s_0 of RPLTS L and a formula ψ . Let $\text{Pr}(s_0, \psi) = (N, E)$ be the dependency graph. We then have the following.*

- If (s, F) is a false node then

$$\llbracket (s, F) \rrbracket = \emptyset \tag{6}$$

- If (s, F) is an empty node then

$$\llbracket (s, F) \rrbracket = \mathcal{M}_L(s) \tag{7}$$

- If (s, F) is an action node then for all $a \in \text{action}(F)$ let $F_a = \text{residue}(F, a)$. Then

$$\llbracket (s, F) \rrbracket = \bigcap_{a \in \text{action}(F)} \bigcup_{((s, F), a, (s', F_a)) \in E} f_{s, a}(\llbracket (s', F_a) \rrbracket) \tag{8}$$

- If (s, F) is a non-action interior node then

$$\llbracket (s, F) \rrbracket = \bigcup_{((s, F), \epsilon^+, (s, F'))} \llbracket (s, F') \rrbracket \tag{9}$$

All equations given above except for Equation 8, relating the semantics of an action node and its successors, are straightforward to verify. Note that an action node (s, F) has for every $a \in \text{action}(F)$ as many a successors in $\text{Pr}(s_0, \psi)$ as there are a successors of the node s in the RPLTS. Note further that $\text{residue}(F, a)$ can be satisfied by any of these a -successors: thus the inner union operator. However, for all $a \in \text{action}(F)$ the formula $\text{residue}(F, a)$ needs to be satisfied; hence the outer intersection operator. We now turn to the proof.

Proof All relationships except the one for action nodes trivially follow from semantics definitions of GPL formulae, and are omitted.

Consider an action node (s, F) . Note that for each $a \in \text{action}(F)$ that (a) $\exists s'.(s, a, s') \in \delta$ and (b) there is at least one of $\langle a \rangle \psi'$ or $[a] \psi'$, for some ψ' , in F . Further, note that every s -rooted d-tree has only one a branch. Thus, we can apply the following reasoning:

$$\begin{aligned}
\llbracket (s, F) \rrbracket &= \Theta_{L,s}(\wedge F) \\
&\quad (\text{by GPL semantics}) \\
&= \left(\bigcap_{\langle a \rangle \psi \in F} \Theta_{L,s}(\langle a \rangle \psi) \right) \cap \left(\bigcap_{[a] \psi \in F} \Theta_{L,s}([a] \psi) \right) \\
&\quad (\text{by definition of } \mathbf{f}_{(s,a)} \text{ and GPL semantics}) \\
&= \left(\bigcap_{\langle a \rangle \psi \in F} \mathbf{f}_{(s,a)}(\Theta_L(\psi)) \right) \cap \left(\bigcap_{[a] \psi \in F} \mathbf{f}_{(s,a)}(\Theta_L(\psi)) \right) \\
&\quad (\text{By Equation 3 in Lemma 2}) \\
&= \bigcap_{\langle a \rangle \psi \in F \vee [a] \psi \in F} \mathbf{f}_{(s,a)}(\Theta_L(\psi)) \\
&\quad (\text{by grouping factors}) \\
&= \bigcap_{a \in \text{action}(F)} \bigcap_{\psi \in \text{residue}(F,a)} \mathbf{f}_{(s,a)}(\Theta_L(\psi)) \\
&\quad (\text{by Lemma 6}) \\
&= \bigcap_{a \in \text{action}(F)} \mathbf{f}_{(s,a)}(\Theta_L(\wedge_{\psi \in \text{residue}(F,a)} \psi)) \\
&\quad (\text{by definition of } \mathbf{f}_{(s,a)}) \\
&= \bigcap_{a \in \text{action}(F)} \bigcup_{(s,a,s') \in \delta} \mathbf{f}_{(s,a)}(\Theta_{L,s'}(\wedge_{\psi \in \text{residue}(F,a)} \psi)) \\
&= \bigcap_{a \in \text{action}(F)} \bigcup_{(s,a,s') \in \delta} \mathbf{f}_{(s,a)}(\llbracket (s', \text{residue}(F, a)) \rrbracket) \\
&= \bigcap_{a \in \text{action}(F)} \bigcup_{((s,F),a,(s',\text{residue}(F,a))) \in E} \mathbf{f}_{(s,a)}(\llbracket (s', \text{residue}(F, a)) \rrbracket)
\end{aligned}$$

□

We can now extend the semantic relationships to measures of the nodes in the graph $\text{Pr}(s_0, \psi)$. Again, as earlier, most of the cases are easily explained except for the measure of the semantics of action nodes.

Lemma 8 *Fix $\text{Pr}(s_0, \psi) = (N, E)$. The measure $\mathbf{m}_s(\llbracket (s, F) \rrbracket)$ of a node (s, F) is as follows:*

- If (s, F) is a true node then

$$\mathbf{m}_s(\llbracket (s, F) \rrbracket) = 1 \tag{10}$$

- If (s, F) is a false node then

$$\mathbf{m}_s(\llbracket (s, F) \rrbracket) = 0 \tag{11}$$

- If (s, F) is an or-node with edges $((s, F), \epsilon^+, (s, F_1))$, $((s, F), \epsilon^+, (s, F_2))$ and $((s, F), \epsilon^-, (s, F_3))$ then

$$\mathbf{m}_s(\llbracket (s, F) \rrbracket) = \mathbf{m}_s(\llbracket (s, F_1) \rrbracket) + \mathbf{m}_s(\llbracket (s, F_2) \rrbracket) - \mathbf{m}_s(\llbracket (s, F_3) \rrbracket) \quad (12)$$

- If (s, F) is an action node then

$$\mathbf{m}_s(\llbracket (s, F) \rrbracket) = \prod_{a \in \mathbf{action}(F)} \sum_{((s, F), a, (s', F')) \in E} P(s, a, s') \times \mathbf{m}_{s'}(\llbracket (s', F') \rrbracket) \quad (13)$$

- If (s, F) is none of the above cases then it has a unique successor (s, F') and in this case:

$$\mathbf{m}_s(\llbracket (s, F) \rrbracket) = \mathbf{m}_s(\llbracket (s, F') \rrbracket) \quad (14)$$

Proof Equation 12 follows from inclusion-exclusion principle. Similarly, Equations 10, 11, and 14 are obvious.

Consider Equation 13. To show this identify let us start with Equation 8, which captures its semantics.

$$\begin{aligned} & \mathbf{m}_s(\llbracket (s, F) \rrbracket) \\ &= \mathbf{m}_s(\bigcap_{a \in \mathbf{action}(F)} (\bigcup_{((s, F), a, (s', F')) \in E} \mathbf{f}_{(s, a)}(\llbracket (s', F') \rrbracket))) \\ & \quad \text{(by independence)} \\ &= \prod_{a \in \mathbf{action}(F)} \mathbf{m}_s(\bigcup_{((s, F), a, (s', F')) \in E} \mathbf{f}_{(s, a)}(\llbracket (s', F') \rrbracket)) \\ & \quad \text{(by inclusion-exclusion principle and determinacy of d-trees)} \\ &= \prod_{a \in \mathbf{action}(F)} \sum_{((s, F), a, (s', F')) \in E} \mathbf{m}_s(\mathbf{f}_{(s, a)}(\llbracket (s', F') \rrbracket)) \\ & \quad \text{(by definition of the measure function } \mathbf{m}_s) \\ &= \prod_{a \in \mathbf{action}(F)} \sum_{((s, F), a, (s', F')) \in E} P(s, a, s') \times \mathbf{m}_{s'}(\llbracket (s', F') \rrbracket) \end{aligned}$$

In this calculation the first two steps deserve additional explanation. The first step follows from the following claim:

Claim 1 *If $a, b \in Act$ and $a \neq b$ then for all $M_1, M_2 \subseteq \mathcal{M}_L$ $\mathbf{m}_s(\mathbf{f}_{(s, a)}(M_1) \cap \mathbf{f}_{(s, b)}(M_2)) = \mathbf{m}_s(\mathbf{f}_{(s, a)}(M_1)) \cdot \mathbf{m}_s(\mathbf{f}_{(s, b)}(M_2))$*

The claim follows from the fact that when $\mathbf{f}_{(s, a)}$ constructs trees, the subtrees of the root with edges labeled by $b \neq a$ are constructed independently.

The second step uses the observation that the set of d-trees obtained by completing the partial tree $\{s \xrightarrow{a} s_1\}$ and the partial tree $\{s \xrightarrow{a} s_2\}$ do not intersect when $s_1 \neq s_2$. Thus, there are no terms necessary to compensate for an

event being counted twice and, hence, no negative terms. Thus, Equation 13 is valid. \square

4.1.2 Generating Equations from the Graph

We now explain how to generate a system of equations from the graph described above. The system will contain one variable, $X_{(s,F)}$, for each node (s, F) in the graph and one equation containing this variable as its left-hand side. The right-hand side of the equation for $X_{(s,F)}$ is generated as follows, based on the edges emanating from (s, F) . Clearly, these are based on Lemma 8.

- (1) If (s, F) is an empty node then the equation for $X_{(s,F)}$ is $X_{(s,F)} = 1$; if (s, F) is a false node, the equation for $X_{(s,F)}$ is $X_{(s,F)} = 0$.
- (2) If there is an edge of the form $((s, F), \epsilon^+, (s, F'))$ then the equation for $X_{(s,F)}$ is

$$X_{(s,F)} = \sum_{((s,F), \epsilon^+, (s,F')) \in E} X_{(s,F')} - \sum_{((s,F), \epsilon^-, (s,F')) \in E} X_{(s,F')}.$$

- (3) If (s, F) is an action node then

$$X_{(s,F)} = \prod_{a \in \mathbf{action}(F)} \sum_{((s,F), a, (s', F')) \in E} (P(s, a, s') \cdot X_{(a, F')}).$$

We now have the following..

Lemma 9 *Let $\vec{E} = \{X_{(s,F)} = E_{(s,F)} \mid (s, F) \in N\}$ be the equations generated above, and let \vec{A} be the vector $\{X_{(s,F)} = \mathbf{m}_s(\Theta_{L,s}(\wedge F)) \mid (s, F) \in N\}$. Then \vec{A} is a solution to \vec{E} .*

4.1.3 Solving the Equations

The previous lemma indicates that the equations we generate are “faithful” to the measures we wish to calculate in the sense that they are indeed a solution to the equations. However, in general there will be many such solutions, and the question then arises as to how we determine which solution indeed corresponds to the measures we want. The procedure *modchk-fuzzy* uses the following subprocedure *Iterate* as follows.

Algorithm *Iterate*

- (1) Compute the strongly connected components of the graph, and order them based on their dependence such that if there is a path from a component C to C' then C appears before C' . Note that such an ordering always exists by definition of strongly connected components.

- (2) Propagate solutions as far as possible; if a solution has been computed for a variable, replace all occurrences of the variable in the right-hand sides by its solution.
- (3) Beginning at the end of the strongly connected component list (i.e., at a strongly connected component from which no other strongly connected component can be reached), process each component C as follows.
 - (a) If C contains a μ -node, assign each variable corresponding to a node in C the value 0; otherwise, assign each variable the value 1.
 - (b) Repeatedly calculate new values for the variables of C by evaluating each right-hand side using the old values. Stop when $|\vec{E}(\vec{x}) - \vec{x}|$, the difference between the left and the right hand sides treated as a vector, is less than $\vec{\epsilon}$, a vector of values containing ϵ .
 - (c) Propagate these values.

In general, this algorithm requires the specification of an “error tolerance” ϵ because the quantities being manipulated are real numbers. So the algorithm is approximation-based. However, all the functions being used are continuous, and hence the iteration process described above converges. We now make that claim formal:

Lemma 10 *Let $s_0 \in S$ and ψ_0 be a fuzzy formula. Consider the processing of a SCC C in Step (3b) of Algorithm Iterate. The sequence of values calculated for X_n , where $n = (s, F) \in C$, converges towards the measure $\mathbf{m}_s(\llbracket n \rrbracket)$. Furthermore, there is an iteration i of the inner loop where the approximants are close to the actual solution by ϵ .*

Proof If the SCC C is trivial then the lemma holds vacuously. Thus, let C be a non-trivial SCC. Now, there are two cases to consider. Either C has a μ node or it only contains ν nodes. We will consider the former now, the latter case is similar.

To show that our iterative algorithm works we will begin with an iterative scheme to calculate $\llbracket (s, F) \rrbracket$, and then show how our calculation of the measure closely follows it. Note that we have a set of equations over variables X_n , $n \in C$. Now consider a set of tree-valued variables T_n , $n \in C$. Based on equations 6–9 we can frame equations that relate the variables T_n , $n \in C$. It should be noted that the edges labelled ϵ^- will not play a part in these tree equations. However, it is easy to see that by applying the measure function to the tree equations we get the polynomial equations (which also takes into account the ϵ^- edges).

Let $n = (s, F)$ be a μ node. This means that $F = F' \cup \{\mu x.\psi\}$, where $\mu x.\psi \notin F'$. Further, $\llbracket (s, F) \rrbracket = \llbracket (s, F') \rrbracket \cap (\cup_{i \geq 0} X^i)$ where

$$X^0 = \emptyset, \forall i \geq 0. X^{i+1} = \llbracket \psi \rrbracket [x \mapsto X^i]$$

This iterative characterization implies that to obtain the correct solution to equations in T_C , and, in particular, to the variable for the μ -node $T_{(s,F)}$ we need to start with an assignment of \emptyset to each of the variables $T_{(s',F')}$, $(s', F') \in C$, and iterate through the equations T_C . That this iteration converges to the right set of trees follows from continuity of operations involved in T_C . Let $T_{(s,F)}^i$ denote the result of the i^{th} iteration in computing a solution for $T_{(s,F)}$. As the operators used in the tree equations are monotonic we have that $T_n^i \subseteq T_n^{i+1}$.

Now consider the numeric equations, which are obtained by applying the measure function to the tree equations. In the inner loop of Algorithm *Iterate* we are, therefore, calculating in iteration j the value $X_n^j = \mathbf{m}_s(T_n^j)$ for every node $n \in C$. Given that the various approximations T_n^j are monotonically increasing the probabilities X_n^j are also monotonically increasing. Finally, since the approximations T_n^j converge to the solution for T_n the numerical approximations X_n^j also converge to their solution. Note that this is notwithstanding the negative terms that appear in the equations corresponding to or-nodes; this is because the value of the negative term always grows slower than the positive terms. The proof for the ν case is similar.

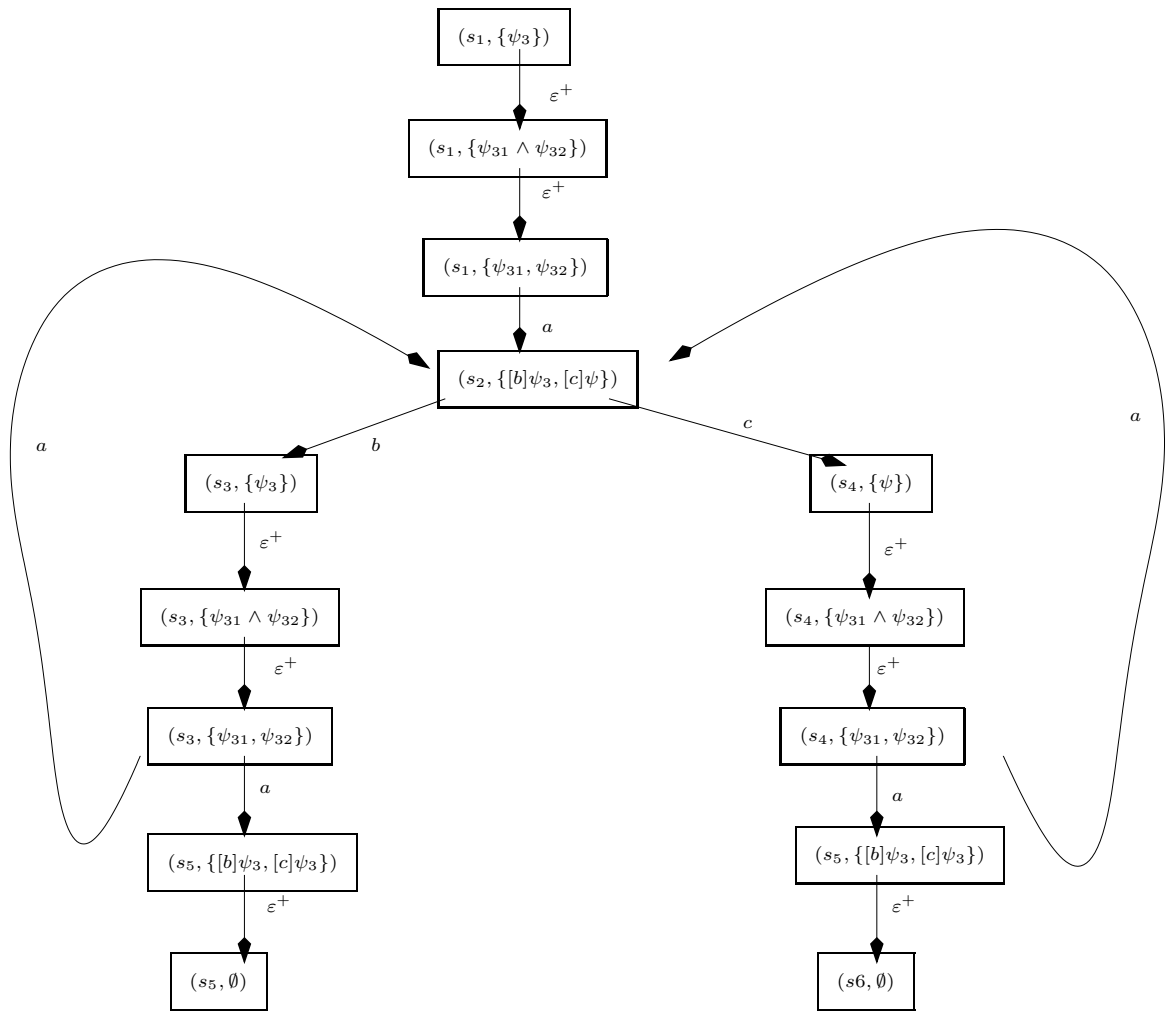
That there is an iteration i where the inner loop of Algorithm *Iterate* terminates follows from the fact that successive iteration produces larger values and the limit of the sequence is indeed the true solution. By the definition of the limit of sequences there is a N after which we are close to a solution by ϵ . \square

It should be noted that our approach works but is not necessarily the most efficient technique. More importantly, the rate of convergence should be better than linear as we are dealing with polynomials. In particular, it is not possible for the difference in successive values obtained in the inner loop to be small but yet the iteration process is far away from the solution that is being sought.

There are better techniques for solving polynomial equations – see, for instance, the work of van Henteryck, McAllester and Kapur [vHMD97] where a robust algorithm *Newton* is presented. *Newton* when given a set of polynomial equations, a tolerance ϵ and a cuboid I in n -dimensional space returns all solutions to the equations present in the cuboid I to within ϵ .

We will illustrate all of the steps used above on Example 2.

Example 4 In Example 2 we claimed that $\psi_3 = \mu X.[a][b]X \wedge [a][c]X$ holds with probability $\frac{1}{9}$ for s_1 rooted trees. We now show how to establish the same information using our model-checking algorithm. The dependency graph $\text{Pr}(s_1, \psi)$ is given in Figure 6 which uses the abbreviations $\psi'_1 = [a][b]\psi$ and $\psi'_2 = [a][c]\psi$. We can read off the following equations from the dependency graph:



Legend: $\psi = \mu X.[a][b]X \wedge [a][c]X$, $\psi_1 = [a][b]\psi$ and $\psi_2 = [a][c]\psi$

Fig. 6. Dependency graph $\text{Pr}(s_1, \psi)$

$X_{(s_1, \{\psi_3\})} = X_{(s_1, \{\psi'_1 \wedge \psi'_2\})}$	$X_{(s_5, \{[b]\psi_3, [c]\psi_3\})} = 1$
$X_{(s_1, \{\psi'_1 \wedge \psi'_2\})} = X_{(s_1, \{\psi_1, \psi'_2\})}$	$X_{(s_6, \{[b]\psi_3, [c]\psi_3\})} = 1$
$X_{(s_1, \{\psi'_1, \psi'_2\})} = X_{(s_2, \{[b]\psi_3, [c]\psi_3\})}$	

$$\begin{aligned}
X_{(s_2, \{[b]\psi_3, [c]\psi_3\})} &= X_{(s_3, \{\psi_3\})} \times X_{(s_4, \{\psi_3\})} \\
X_{(s_3, \{\psi_3\})} &= X_{(s_3, \{\psi'_1 \wedge \psi'_2\})} \\
X_{(s_3, \{\psi'_1 \wedge \psi'_2\})} &= X_{(s_3, \{\psi'_1, \psi'_2\})} \\
X_{(s_4, \{\psi_3\})} &= X_{(s_4, \{\psi'_1 \wedge \psi'_2\})} \\
X_{(s_4, \{\psi'_1 \wedge \psi'_2\})} &= X_{(s_4, \{\psi'_1, \psi'_2\})} \\
X_{(s_3, \{\psi'_1, \psi'_2\})} &= \frac{3}{4}X_{(s_2, \{[b]\psi_3, [c]\psi_3\})} + \frac{1}{4}X_{(s_5, \{[b]\psi_3, [c]\psi_3\})} \\
X_{(s_4, \{\psi'_1, \psi'_2\})} &= \frac{3}{4}X_{(s_2, \{[b]\psi_3, [c]\psi_3\})} + \frac{1}{4}X_{(s_6, \{[b]\psi_3, [c]\psi_3\})}
\end{aligned}$$

Note that the second group of equations form a SCC, while all other equations form singleton SCCs. With back substitution we can see that these equations reduce to

$$\begin{aligned}
x_1 &= x_2 \times x_3 \\
x_2 &= \frac{3}{4}x_1 + \frac{1}{4} \\
x_3 &= \frac{3}{4}x_1 + \frac{1}{4}
\end{aligned}$$

where $x_1 = X_{(s_2, \{[b]\psi_3, [c]\psi_3\})}$, $x_2 = X_{(s_3, \{\psi_3\})}$ and $x_3 = X_{(s_4, \{\psi_3\})}$. Starting with a 0 for each of the three variables we get the following sequence of values for x_1 , which tends towards the solution $\frac{1}{9}$:

0.0625003
0.0881353
0.0999203
0.1055863
0.1083663
0.1097433
0.1104283
0.1107703
0.1109413
0.1110263
0.1110683
0.1110903
0.1111003
0.1111063
0.1111083
0.1111103
0.1111103

This leads to the solution of $\frac{1}{9}$ for the variable $X_{(s_1, \{\psi_3\})}$ – a solution we es-

tablished in Example 2.

In Example 3 we also considered the formula ψ_4 which is similar to ψ_3 except that the outer operator was a ν rather than a μ . If we build a dependency graph for ψ_4 it would be isomorphic to the dependency graph for ψ_3 except that all occurrences of ψ_3 in Figure 6 would be replaced by ψ_4 . The equations derived from the new dependency graph would be identical to what we just considered. The only difference is that we would now be looking for the maximal solution to the equations given above. The solution would be 1 which indeed agrees with the result from Example 3.

4.2 Model Checking and GPL

The procedure *modchk-fuzzy* may now be used to build a model-checker for GPL. This model checker engages in a case analysis on the formula ϕ and performs the obvious operations if the formula is not of the form $\Pr_{\geq p}\psi$ or $\Pr_{> p}\psi$. In these latter two cases, *modchk-fuzzy* is called to calculate $\mathbf{m}_s(\Theta_{L,s}(\psi))$, and the answer compared to p appropriately. As *modchk-fuzzy* is an approximation-based numerical algorithm, the usual numerical issues must be confronted in performing these comparisons. In particular, if the computed answer is close enough to p to fall within the margin of error, then only indeterminate answers can be given.

4.3 Discussion

The algorithm just described relies on the use of numerical approximation techniques. However, in certain cases exact solutions can be calculated. For example, if the RPLTS is in fact a Markov Process then the equation system generated is linear, as in Equation 13 we have $|\mathbf{action}(F) = 1|$. In addition, results of [CY88] suggest that this linear system has an unique solution. In this case, the equations can be solved exactly.

5 Probabilistic Bisimulation

In this section we will show that a recursion-free version of GPL characterizes probabilistic bisimulation for RPLTS. The recursion-free version $\text{GPL}_{\mathbf{f}}$ is as

follows:

$$\begin{aligned}\phi &::= A \mid \neg A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \text{Pr}_{>r}\psi \mid \text{Pr}_{\geq r}\psi \\ \psi &::= \phi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \langle a \rangle \psi \mid [a]\psi\end{aligned}$$

We restrict the set $Prop$ from which A is drawn to be $\{\mathbf{tt}, \mathbf{ff}\} \cup \{\Delta_a \mid a \in Act\}$. The interpretations of these atomic propositions may be found in Section 3. In what follows we will use $\Psi_{\mathbf{f}}$ and $\Phi_{\mathbf{f}}$ to refer to the recursion-free GPL and fuzzy formulae, respectively.

To define the notion of probabilistic bisimulation [LS91] shows how to lift an equivalence relation on S to probability distributions over S . Every equivalence relation R , on S , drives an equivalence relation R^* on distributions over S such that two distributions are related provided they agree on the probability they assign to each equivalence class of R .

Definition 13 *Let $R \subseteq S \times S$ be an equivalence relation. Define the equivalence relation $R^* \subseteq (S \rightarrow [0, 1]) \times (S \rightarrow [0, 1])$ as follows:*

$$\wp_1 R^* \wp_2 \text{ iff } \forall B \in S/R. \wp_1(B) = \wp_2(B)$$

Bisimulation for labeled transition systems dictates that two states are bisimilar provided they can each carry out the action that the other can do and arrive at bisimilar states. Probabilistic bisimulation is similar except that related target states of transitions are replaced by related target distributions of transitions. Formally,

Definition 14 *Fix RPLTS $L = (S, \delta, P, I)$. An equivalence relation $R \subseteq S \times S$ is a probabilistic bisimulation provided if $(s_1, s_2) \in R$ then (a) $I(s_1) = I(s_2)$ and (b) for all $a \in Act$ we have $(\lambda s'_1. P(s, a, s'_1)) R^* (\lambda s'_2. P(s_2, a, s'_2))$.*

Two states s_1 and s_2 are bisimilar iff there is a bisimulation R which includes them. Furthermore, we will use \sim to denote the largest bisimulation.

In the following we will first show that probabilistically bisimilar states satisfy the same set of recursion-free, quantifier-free fuzzy GPL formulae. To achieve this result we appeal to the the dependency graph that we construct for calculating measures. To that end, we will make a RPLTS out of such dependency graphs.

Definition 15 *Given a RPLTS $L = (S, \delta, P, I)$, a pair of states $s_1, s_2 \in S$, and a formula $\psi \in \Psi_{\mathbf{f}}$, let $(N_1, E_1) = \text{Pr}(s_1, \psi)$ and $(N_2, E_2) = \text{Pr}(s_2, \psi)$. Then define RPLTS $\text{Pr}_L(s_1, s_2, \psi) = (N, E, P_{s_1, s_2, \psi}, I_{s_1, s_2, \psi})$, where $N = N_1 \cup N_2$,*

$E = E_1 \cup E_2$, and

- $I_{s_1, s_2, \psi}(s, F) = I(s)$,
- $P_{s_1, s_2, \psi}((s, F), a, (s', F')) = \begin{cases} P(s, a, s') & \text{if } a \in Act \\ \frac{1}{k} & \text{if } a = \epsilon^+ \text{ and } k = |\{(s, F), \epsilon^+, (s', F')\}| \\ 1 & \text{otherwise, i.e. } a = \epsilon^- \end{cases}$

It should be noted that because of the absence of recursion dependency graph $\text{Pr}_L(s_1, s_2, \psi)$ will always be acyclic. Let the depth of a node (t, F) in the graph $\text{Pr}_L(s, \psi)$ be the length of longest path from (t, F) to any sink node. We will now show that there is a bisimulation relation between the states of the RPLTS induced by the dependency graph defined above.

Lemma 11 *Let L be a RPLTS, let $s_1 \sim s_2$, and let $\psi \in \Psi_{\mathbf{f}}$. Furthermore, let $\text{Pr}_L(s_1, s_2, \psi) = (N, E, P_{s_1, s_2, \psi}, I_{s_1, s_2, \psi})$. Then the relation*

$$R = \{((t, F), (u, F)) \mid (t, F), (u, F) \in N, t \sim u\}$$

is a probabilistic bisimulation.

Proof The proof proceeds by induction on the depth of a node (t, F) in the graph defined by (N, E) . The case analysis in turn examines the form of the transition labels emanating from (t, F) . Most cases are routine; we consider in detail the case when $((t, F), a, (t', F'))$ and $a \in Act$. Clearly, (u, F) can only have transitions labeled by actions, since the formula sets in both (t, F) and (u, F) are identical. In addition, $((t, F), a, (t', F')) \in P_{s_1, s_2, \psi}$ requires that $t \xrightarrow{a} t'$ in L , and as $t \sim u$ it follows that u has an a -transition in L as well and that t and u assign the same probabilities to the equivalence classes in R containing the targets of their a -transitions. The result therefore follows. \square

The next lemma allows us to prove that the measures of fuzzy $\text{GPL}_{\mathbf{f}}$ formulae are identical for bisimilar states of a RPLTS.

Lemma 12 *Let L be a RPLTS. If $s_1 \sim s_2$ then for all quantifier-free fuzzy $\text{GPL}_{\mathbf{f}}$ formulae ψ we have*

$$\mathbf{m}_{s_1}(\Theta_{L, s_1}(s_1, \{\psi\})) = \mathbf{m}_{s_2}(\Theta_{L, s_2}(s_2, \{\psi\}))$$

Proof Consider the RPLTS $\text{Pr}_L(s_1, s_2, \psi)$, which has the dependency graphs $\text{Pr}(s_1, \psi)$ and $\text{Pr}(s_2, \psi)$ embedded in it. We now prove by induction on the depth, m , of (t, F) in this RPLTS that $\mathbf{m}_t(\llbracket (s_1, F) \rrbracket) = \mathbf{m}_u(\llbracket (s_2, F) \rrbracket)$ provided that $t \sim u$. This stronger statement implies the desired result.

Base case $m = 0$. In this case (t, F) is an empty node or it is a false node. Since (t, F) is an empty node if and only if (u, F) is an empty node (cf. Lemma 11), they agree on the measure.

Induction case $m > 0$. Assume hypothesis true for $m = k$ and consider the case of $m = k+1$. We now carry out a case analysis on the form of the outgoing transitions from (t, F) and (u, F) . Note that the transitions in both cases must either consist of a single transition labeled by ϵ^+ , or of two transitions labeled by ϵ^+ and by ϵ^- , or by actions, since the formula sets are the same, F . In the first two cases the result follows immediately from the induction hypothesis. In the last case, the following calculations show that the measures are the same.

$$\begin{aligned}
\mathbf{m}_t(\llbracket(t, F)\rrbracket) &= \prod_{a \in \text{action}(F)} \sum_{t'.(t,a,t') \in \delta} P(t, a, t') \times \mathbf{m}_{t'}(\llbracket(t', F_a)\rrbracket) \\
&\quad (\text{by definition}) \\
&= \prod_{a \in \text{action}(F)} \sum_{B \in S/\sim} \sum_{t' \in B} P(t, a, t') \times \mathbf{m}_{t'}(\llbracket(t', F_a)\rrbracket) \\
&\quad (\text{by induction hypothesis}) \\
&= \prod_{a \in \text{action}(F)} \sum_{B \in S/\sim} \mathbf{m}_{[B]}(\llbracket([B], F_a)\rrbracket) \times \sum_{t' \in B} P(t, a, t') \\
&\quad (\text{using } [B] \in B, \text{ by definition of probabilistic bisimulation}) \\
&= \prod_{a \in \text{action}(F)} \sum_{B \in S/\sim} \sum_{u' \in B} P(u, a, u') \times \mathbf{m}_{u'}(\llbracket(u', F_a)\rrbracket) \\
&= \prod_{a \in \text{action}(F)} \sum_{B \in S/\sim} \sum_{u'.(u,a,u') \in \delta} P(u, a, u') \times \mathbf{m}_{u'}(\llbracket(u', F_a)\rrbracket)
\end{aligned}$$

□

Lemma 12 can now be extended to all $\text{GPL}_{\mathbf{f}}$ formulae, which can be easily proved by induction on the structure of formulae:

Theorem 4 *Fix RPLTS L . If $s_1 \sim s_2$ then for all $\phi \in \Phi_{\mathbf{f}}$ we have $s_1 \models_L \phi$ iff $s_2 \models_L \phi$.*

From Theorems 2 and 4 we get the main theorem of this section.

Theorem 5 *Let L be a RPTLS with states s_1 and s_2 . Then $s_1 \sim s_2$ iff, for all $\phi \in \Phi_{\mathbf{f}}$, $s_1 \models_L \phi$ iff $s_2 \models_L \phi$.*

This result may be extended to full GPL (i.e. with recursion) without difficulty; the details are omitted.

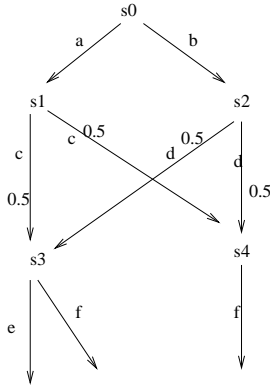


Fig. 7. Example due to de Alfaro

6 Concluding Remarks

We have presented a uniform framework for defining temporal logics on reactive probabilistic transition systems. Our approach is based on using the modal mu-calculus to define measurable sets of observations of such systems. We have showed that our logic is expressive enough to three very different existing temporal logics. A model-checking procedure for the logic that relies on solving non-linear equations was presented, and the logic was shown to characterize probabilistic bisimulation.

It should be emphasized that in the reactive model the nonprobabilistic choices are *external*: the environment, not the system itself, selects which action to perform. This point of view stands in contrast to models like Markov decision processes, in which nonprobabilistic choices are internal. To see the difference, consider the following example raised by Luca de Alfaro in the context of the CTL formula $EF\neg\Delta_e$, which asserts that there exists a path where eventually an e is possible.

In the Markov decision process model, as epitomized by [BdA95], schedulers are used to resolve nondeterministic choices, with the resulting Markov processes then used to interpret formulae. In this example, irrespective of which path (either a or b) is chosen by the scheduler the probability of eventually being able to do an e is $\frac{1}{2}$. However, our observation-based approach would say that the probability is $\frac{3}{4}$, since three of the four trees obtained by resolving the two (independent, uniform) probabilistic choice points satisfy the formula. Clearly, these two different answers rely on different assumptions – that of external nondeterminism and branching-time semantics in our approach, and that of internal nondeterminism and linear-time semantics in the scheduler approach.

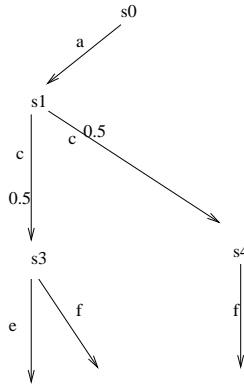


Fig. 8. An example of RPLTS

As for future work, we believe that we can improve on the algorithm presented here by using results similar to those in [vHMD97] which provides a robust and efficient method for solving systems of polynomial equations based on interval arithmetic. Another important issue for future work is that of applying our logic to more general transition systems (for example, the transition systems of [Seg95] which include both external and internal choice) and establishing its relation to probabilistic automata [Paz71]. Furthermore, it would be interesting to work out probabilistic model-checking algorithm for the general case. Finally, it would also be useful to investigate the adaptation of our techniques to models of distributed computation in which resources may probabilistically fail, such as the one presented in [PSC⁺98].

Acknowledgments: Murali Narasimha would like to thank S. Arun-Kumar and E. Kaltofen for several helpful discussions on this topic.

References

- [ACD91] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking for probabilistic real-time systems. In *Automata, Languages and Programming: Proceedings of the 18th ICALP*, volume 510 of *LNCS*, pages 115–136, 1991.
- [And94] H.R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, April 1994.
- [ASB⁺95] A. Aziz, V. Singhal, F. Balarin, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *Proc of Computer Aided Verification '95*. Springer-Verlag, Jul 1995.

- [BC96] G. Bhat and R. Cleaveland. Efficient local model checking for fragments of the modal μ -calculus. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS '96)*, volume 1055, pages 107–126, Passau, Germany, March 1996.
- [BdA95] A. Bianco and L. de Alfaro. Model-checking of probabilistic and non-deterministic systems. In *Proc. Foundations of software technology and theoretical computer science*, Lecture notes in Computer science, vol 1026, pages 499–513. Springer-Verlag, December 1995.
- [BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.
- [BPS01] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Amsterdam, 2001.
- [BS01] E. Bandini and R. Segala. Axiomatizations for probabilistic bisimulation. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2001.
- [CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.
- [Cle90] Rance Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27(8):725–747, 1990.
- [CS93] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. 2:121–147, 1993.
- [CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Proc. 1988 IEEE Symp. on the Foundations of Comp. Sci.*, 1988.
- [DH83] R. De Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1983.
- [EC80] E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs as fixpoints. In J. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming (ICALP '80)*, volume 85, pages 169–181, Utrecht, July 1980.
- [EH86] E. A. Emerson and J. Y. Halpern. “sometimes” and “not never” revisited: On branching time versus linear time temporal logic. *JACM*, 33(1):151–178, 1986.
- [EL86] E. Allen Emerson and Chin-Laung Lei. Efficient model-checking in fragments of the propositional mu-calculus. In *Proceedings 1st Annual IEEE Symp. on Logic in Computer Science, LICS'86, Cambridge, MA, USA, 16–18 June 1986*, pages 267–278. IEEE Computer Society Press, Los Alamitos, CA, 1986.

- [Han94] H. Hansson. *Time and Probability in Formal Design of Distributed Systems*. Elsevier, 1994.
- [HK97] Michael Huth and Marta Kwiatkowska. Quantitative analysis and model checking. In *Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 111–122, Warsaw, Poland, 29 June–2 July 1997. IEEE Computer Society Press.
- [HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the Association of Computing Machinery*, 32(1):137–161, 1985.
- [Koz83] D. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science*, 27(1):333–354, 1983.
- [KSK66] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. Van Nostrand, 1966.
- [KVV00] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.
- [LBCJ94] D. E. Long, A. Browne, E. M. Clarke, and S. Jha. An improved algorithm for the evaluation of fixpoint expressions. *Lecture Notes in Computer Science*, 818:338–??, 1994.
- [LS91] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94, 1991.
- [McM93] McMillan, K. L. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell Massachusetts, 1993.
- [Mil89] R. Milner. *Communication and Concurrency*. PHI Series in Computer Science. Prentice Hall, 1989.
- [Nar99] M. Narasimha. *Probabilistic Verification Through Temporal Logics*. PhD thesis, North Carolina State University, 1999.
- [Paz71] Azaria Paz. *Introduction to Probabilistic Automata*. Academic Press, New York, 1971.
- [PSC⁺98] A. Philippou, O. Sokolsky, R. Cleaveland, I. Lee, and S. Smolka. Probabilistic resource failure in real-time process algebra. *Lecture Notes in Computer Science*, 1466, 1998.
- [PZ93] Amir Pnueli and Lenore D. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, March 1993.
- [QS82] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. 5th Int’l Symp. on Programming*, Lecture Notes in Computer Science, Vol. 137, pages 337–371. Springer-Verlag, Berlin/New York, 1982.

- [Seg95] R. Segala. A compositional trace-based semantics for probabilistic automata. In *CONCUR 95*, pages 324–338, 1995.
- [SL95] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, Summer 1995.
- [Sti92] C. Stirling. Modal and temporal logics. In S. Abramsky, D. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 477–563. Oxford University Press, 1992.
- [Var85] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *IEEE Symposium on Foundations of Computer Science*, pages 327–338, 1985.
- [vGSST90a] Rob van Glabbeek, Scott A. Smolka, Bernhard Steffen, and Chris M. N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 130–141, Philadelphia, Pennsylvania, 4–7 June 1990. IEEE Computer Society Press.
- [vGSST90b] Rob van Glabbeek, Scott A. Smolka, Bernhard Steffen, and Chris M. N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proc. of Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 130–141. IEEE Computer Society Press, June 1990.
- [vHMD97] P. van Henteryck, D. McAllester, and D.Kapur. Solving Polynomial Equation Systems Using a Branch and Prune Search. *SIAM Journal of Computing*, 34(2), 1997.