Probabilistic Temporal Logics via the Modal Mu-Calculus

Murali Narasimha Rance Cleaveland Purush Iyer
Department of Computer Science
North Carolina State Uniersity
Raleigh, NC 27695-7534
Email: {mnarasi,rance,purush}@eos.ncsu.edu

Abstract

This paper presents a mu-calculus-based modal logic for describing properties of probabilistic labeled transition systems (PLTSs) and develops a model-checking algorithm for determining whether or not states in finite-state PTLSs satisfy formulas in the logic. The logic is based on the distinction between (probabilistic) "systems" and (non-probabilistic) "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.

1 Introduction

Classical temporal-logic model checking [CES86, McM93] provides a basis for automatically checking the correctness of finite-state systems such as hardware designs and communication protocols. In this framework, systems are modeled as *transition systems*, and requirements are posed as formulas in temporal logic. A model checker then accepts two inputs, a transition system and a temporal formula, and returns "true" if the system satisfies the formula and "false" otherwise.

In traditional model checking, system models include information about the possible choices of execution steps in any given state. The corresponding temporal logics 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 [Koz83, EH86]. When system models include *probabilistic* information regarding their operational behavior, 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. Several examples of applying probabilistic model-checking to practical situations have been reported by Hansson [Han94]. Such motivations have led to the study of numerous probabilistic variants of temporal logic and model checking [ASB$^+$95, BdA95, CY88, Han94, HK97, LS91, PZ93, Var85].

The goal of this paper is to develop a uniform framework for temporal logics for probabilistic systems. To this end, we show how the unifying classical temporal logic, the *modal mu-calculus* [Koz83, EL86], may be altered by adding probabilistic quantifiers constraining the "probability" with which probabilistic systems satisfy mu-calculus formulas. We then show how a variety of existing probabilistic logics may be represented in our framework and present a model-checking algorithm.

2 Probabilistic Transition Systems and the Logic GPL

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

2.1 Reactive Probabilistic Labeled Transition Systems

We use the *reactive probabilistic labeled transition systems* (PLTS for short) of [vGSST90, LS91] as models of probabilistic computation. These are defined with respect to fixed sets *Act* and *Prop* of atomic *actions* and *propositions*, 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 2.1 A PLTS L is a tuple (S, δ, P, I), where

- $(s, s', s_1 \in) S$ is a countable set of states;
- $\delta \subseteq S \times \mathit{Act} \times S$ is the transition relation;
- $P : \delta \to (0, 1]$, the transition probability distribution, satisfies: $\sum_{(s,a,s') \in \delta} P(s, a, s') \in \{0, 1\}$ for all $s \in S$, $a \in \mathit{Act}$; and
- $I : S \to 2^{\mathit{Prop}}$ is the interpretation function.

Intuitively, a PLTS records the operational behavior of a system, with S representing the possible system states and δ the execution steps enabled in different system states; each such step is labeled with an action, and the intention is that when the environment of the system enables the action, the system may engage in a transition labeled by the action. When this is the case, $P(s, a, s')$ 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 $\sum_{(s,a,s') \in \delta} P(s, a, s') = 1$. In what follows we write $s \xrightarrow{a} s'$ if $(s, a, s') \in \delta$.

In this paper we wish to view a (state in a) PTLS 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. More specifically, given a state in the PLTS we can unroll the PLTS 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. Figure 1 presents a sample PLTS, its unrolling from a given state, and an associated observation.

2.2 Syntax of GPL

Generalized Probabilistic Logic (GPL) is parameterized with respect to a set $(X, Y \in) \mathit{Var}$ of propositional variables, a set $(a, b \in) \mathit{Act}$ of actions, and a set $(A \in) \mathit{Prop}$ be a set of atomic propositions. The

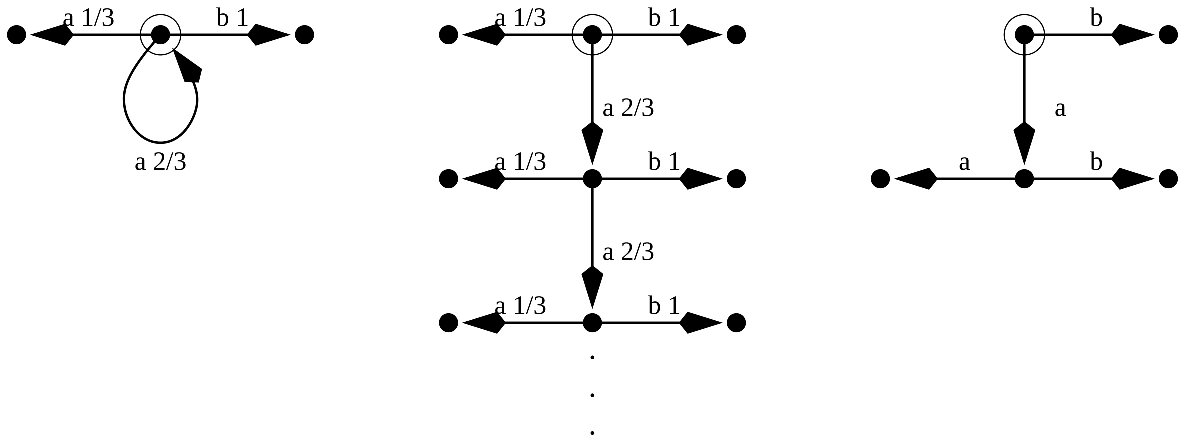

Figure 1: A PLTS, its unrolling from a state, and an observation.

syntax of GPL may then be given using the following BNF-like grammar, where $0 \leq p \leq 1$.

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

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 formulas are required to satisfy the following additional restrictions: they must contain no free variables, and no sub-formula of the form $\mu\, X.\psi$ ($\nu\, X.\psi$) may contain a free ν-occurrence (μ-occurrence) of a variable.[1] In what follows we refer to formulas generated from nonterminal ϕ etc. as *state formulas* and those generated from ψ as *fuzzy formulas*; the formulas of GPL are the state formulas. We use $(\phi, \phi' \in)\Phi$ to represent the set of all state formulas and $(\psi, \psi' \in)\Psi$ for the set of all fuzzy formulas. In the remainder of the paper we write $\gamma[\Gamma'/X]$ to denote the 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 formulas are to be interpreted as specifying sets of *observations* of PLTSs, which are themselves non-probabilistic 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 an observation satisfying ψ, while it satisfies $[a]\psi$ if every a-transition leads to such an observation. 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 PLTS 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 sense. State formulas will then be interpreted with respect to states in PLTSs, with a state satisfying a formula of the form $Æ_{\geq p}\psi$ if the measure of observations corresponding to the state is at least p.

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

2.3 Semantics of GPL

This subsection formalizes of the notions described informally above. We first define observations of a PLTS and show how the observations from a given state in a PLTS form a probability space. We then use these probability spaces to interpret GPL formulas. In what follows we fix sets *Act* and *Prop*.

2.3.1 PLTSs and Measure Spaces of Observations

To define the observation trees of a PLTS we introduce *partial computations*, which will form the nodes of the trees.

Definition 2.2 Let $L = (S, \delta, P, I)$ be a PLTS. Then a sequence of the form $s_0 \stackrel{a_1}{\rightarrow} s_1 \cdots \stackrel{a_n}{\rightarrow} s_n$ is a *partial computation* of L if $n \geq 0$ and for all $0 \leq i < n$, $s_i \stackrel{a_{i+1}}{\rightarrow} s_{i+1}$.

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

Definition 2.3 Let $\sigma = s_0 \stackrel{a_1}{\rightarrow} s_1 \cdots \stackrel{a_n}{\rightarrow} s_n$ and $\sigma' = s'_0 \stackrel{a'_1}{\rightarrow} s'_1 \cdots \stackrel{a_{n'}}{\rightarrow} s'_{n'}$ be partial computations of PLTS $L = (S, \delta, P, I)$, and let $a \in Act$.

1. If $s_n \stackrel{a}{\rightarrow} s'_0$ then $\sigma \stackrel{a}{\rightarrow} \sigma'$ is the partial computation $s_0 \stackrel{a_1}{\rightarrow} s_1 \cdots \stackrel{a_n}{\rightarrow} s_n \stackrel{a}{\rightarrow} s'_0 \stackrel{a'_1}{\rightarrow} s'_2 \cdots \stackrel{a_{n'}}{\rightarrow} s'_{n'}$.
2. σ' is a *prefix* of σ if $\sigma' = s_0 \stackrel{a_1}{\rightarrow} s_1 \cdots \stackrel{a_i}{\rightarrow} s_i$ for some $i \leq n$.

We also introduce the following terminology for sets of partial computations.

Definition 2.4 Let $L = (S, \delta, P, I)$ be a PLTS, and let $C \subseteq \mathcal{C}_L$ be a set of computations.

1. C is *prefix-closed* if, for every $\sigma \in C$ and σ' a prefix of σ, $\sigma' \in C$.
2. C is *deterministic* if for every $\sigma, \sigma' \in C$ with $\sigma = s_0 \stackrel{a_1}{\rightarrow} s_1 \cdots \stackrel{a_n}{\rightarrow} s_n \stackrel{a}{\rightarrow} s \cdots$ and $\sigma' = s_0 \stackrel{a_1}{\rightarrow} s_1 \cdots \stackrel{a_n}{\rightarrow} s_n \stackrel{a'}{\rightarrow} s' \cdots$, 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 PLTS L as follows.

Definition 2.5 Let $L = (S, \delta, P, I)$ be a PLTS. 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 C is a d-tree then we use $\mathsf{root}(C)$ to refer to the s such that $C \subseteq \mathcal{C}_L(s)$ and $\mathsf{edges}(C)$ to refer to the relation $\{(\sigma, a, \sigma') \mid \sigma, \sigma' \in C \wedge \exists s' \in S.\sigma' = \sigma \xrightarrow{a} s\}$.

We use $\mathcal{T}_L$ to refer to all the d-trees of L and set $\mathcal{T}_L(s) = \{T \in \mathcal{T}_L \mid \mathsf{root}(T) = s\}$. We call T' a *prefix* of T if $T' \subseteq T$ We write $T \xrightarrow{a} T'$ if $\{\mathsf{root}(T) \xrightarrow{a} \sigma' \mid \sigma' \in T'\} \subseteq T$; 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 PLTS as the "outcomes" of the PLTS 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 $L = (S, \delta, P, I)$.

A *basic cylindrical subset* of $\mathcal{M}_L(s)$ contains all trees sharing a given finite prefix.

Definition 2.6 Let $s \in S$, and let $T \in \mathcal{T}_L(s)$ be finite. Then $B_T \subseteq \mathcal{M}_L(s)$ 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 2.7 Let $T \in \mathcal{T}_L(s)$ be finite, and let B_T be the associated basic cylindrical set. Then the *measure*, $\mathsf{m}(B_T)$, of B_T is given by: $\mathsf{m}(B_T) = \Pi_{(\sigma,a,\sigma') \in \mathsf{edges}(T)} P(\mathsf{lst}(\sigma), a, \mathsf{lst}(\sigma'))$.

Intuitively, $\mathsf{m}(B_T)$ represents the proportion of all maximal d-trees emanating from the root of B_T that have B_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 $\mathsf{root}(T) = s$. We can then define a probability space $(\mathcal{M}_L(s), \mathcal{B}_s, \mathsf{m}_s)$ as follows.

Definition 2.8 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. $\mathsf{m}_s : \mathcal{B}_s \to [0,1]$ is then defined inductively as follows.

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

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

2.3.2 Semantics of Fuzzy Formulas

In the remainder of this section we define the semantics of GPL formulas with respect to a fixed PLTS $L = (S, \delta, P, I)$ by giving mutually recursive definitions of a relation $\models_L \subseteq S \times \Phi$ and a function $\Theta_L : \Psi \to 2^{\mathcal{M}_L}$. 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. In this subsection we present Θ_L; the next subsection then considers $\models_L$. In what follows we fix $L = (S, \delta, P, I)$.

Our intention in defining $\Theta_L(\psi)$ is that it return trees that, 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 2.9 The function Θ_L is defined inductively as follows.

- $\Theta_L(\phi)e = \cup_{s \models_L \phi} \mathcal{M}_L(s)$
- $\Theta_L(X)e = e(X)$
- $\Theta_L(\langle a \rangle \psi)e = \{T \mid \exists T' : T \xrightarrow{a} T' \wedge T' \in \Theta_L(\psi)e\}$
- $\Theta_L([a]\psi)e = \{T \mid (T \xrightarrow{a} T') \Rightarrow T' \in \Theta_L(\psi)e\}$
- $\Theta_L(\psi_1 \wedge \psi_2)e = \Theta_L(\psi_1)e \cap \Theta_L(\psi_2)e$
- $\Theta_L(\psi_1 \vee \psi_2)e = \Theta_L(\psi_1)e \cup \Theta_L(\psi_2)e$
- $\Theta_L(\mu X.\psi)e = \cup_{i=0}^{\infty} M_i$, 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 formulas as mu-calculus formulas 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 close this section by remarking 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 have the following.

Theorem 2.10 *For any $s \in S$ and $\psi \in \Psi$, $\Theta_{L,s}(\psi)$ is measurable.*

2.3.3 Semantics of State Formulas

We now define the semantics of state formulas by defining the relation $\models_L$.

Definition 2.11 Let $L = (S, \delta, P, I)$ be a PLTS. Then $\models_L$ is defined inductively as follows.

- $s \models_L^e A$ iff $A \in I(s)$.
- $s \models_L^e \neg A$ iff $A \notin I(s)$.
- $s \models_L^e \phi_1 \wedge \phi_2$ iff $s \models \phi_1$ and $s \models \phi_2$.

- $s \models_L^e \phi_1 \vee \phi_2$ iff $s \models \phi_1$ or $s \models \phi_2$.
- $s \models_L^e Æ_{>p}\psi$ iff $\mathsf{m}_s(\Theta_{L,s}(\psi)e) > p$.
- $s \models_L^e Æ_{\geq p}\psi$ iff $\mathsf{m}_s(\Theta_{L,s}(\psi)e) \geq p$.

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 $Æ_{>p}\psi$ iff the measure of the observations of ψ rooted at s exceeds p, and similarly for $Æ_{\geq p}\psi$.

2.3.4 Properties of the Semantics

We close this section by remarking on some of the properties of GPL. The first shows that the modal operators for fuzzy formulas enjoy certain distributivity laws with respect to the propositional operators.

Lemma 2.12 *For a PLTS L, fuzzy formulas ψ_1 and ψ_2 and $a \in Act$, 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. $\Theta_L([a]\psi_1 \wedge \langle a\rangle\psi_2) = \Theta_L(\langle a\rangle(\psi_1 \wedge \psi_2))$

That $[a]$ distributes over $\vee$ and $\langle a\rangle$ over $\wedge$ is due to the determinacy of d-trees.

Based on Theorem 2.10 and the definition of Θ_L, the next lemma also holds.

Lemma 2.13 *Let $s \in S$, $a \in Act$ and $\psi, \psi_1, \psi_2 \in \Psi$. Then we have the following.*

$$\mathsf{m}_s(\Theta_L(\psi_1 \vee \psi_2)) = \mathsf{m}_s(\Theta_L(\psi_1)) + \mathsf{m}_s(\Theta_L(\psi_2)) - \mathsf{m}_s(\Theta_L(\psi_1 \wedge \psi_2)) \quad (1)$$

$$\mathsf{m}_s(\Theta_L(\langle a\rangle\psi)) = \sum_{(s,a,s')\in\delta} P(s,a,s') * \mathsf{m}_{s'}(\Theta_L(\psi)) \quad (2)$$

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

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

Lemma 2.14 *Let $L = (S, \delta, P, I)$ be a PLTS with $s \in S$, and let ψ and ϕ be fuzzy and state formulas, respectively. Then there exist formulas $\mathsf{neg}(\psi)$ and $\mathsf{neg}(\phi)$ such that:*

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

Proof Follows from the duality of $\wedge/\vee$, $[a]/\langle a\rangle$, ν/μ, and $Æ_{>p}/Æ_{\geq 1-p}$. ■

3 Expressiveness of GPL

In this section we illustrate the expressive power of GPL by showing how three quite different probabilistic logics may be encoded within it.

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 PLTSs. The formulas of the logic are generated by the following grammar:

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

where $0 \leq p < 1$, $A \in \mathit{Prop}$ and $a \in \mathit{Act}$. Formulas are interpreted with respect to states in a given PLTS $L = (S, \delta, P, I)$ via a relation $\models_L^{PML} \subseteq S \times \phi$. The definition appears below; the cases for $\neg$ and $\wedge$ have been omitted.

$$\begin{array}{ll} s \models_L^{PML} A & \text{iff } A \in I(s) \\ s \models_L^{PML} \langle a\rangle_p \phi & \text{iff } \sum_{\{s' \mid (s,a,s') \in \delta \wedge s' \models_L^{PML} \phi\}} P(s,a,s') \geq p \end{array}$$

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 formulas into GPL formulas.

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

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

Theorem 3.1 *Let ϕ be a PML formula and s be a state of PLTS L. Then $s \models_L^{PML} \phi$ iff $s \models E_{PML}(\phi)$.*

3.2 Encoding pCTL*

pCTL* [ASB+95] represents 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 structures referred to in [ASB+95] as *Markov processes* (MP), which may be viewed as probabilistic Kripke structures. It turns out that MPs form a subclass of PLTSs. This section will show that pCTL* has a uniform encoding in GPL.

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

Definition 3.2 Let $\mathit{Act} = \{a\}$. Then a Markov process (MP) is a PLTS (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 $\pi = s_0 s_1 \ldots$ coincides with the d-tree $\{\sigma_0, \sigma_1, \ldots\}$, 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 $\pi = s_0 s_1 \ldots$: $\pi[i] = s_i$, and $\pi^i = s_i s_{i+1} \ldots$.

3.2.1 Interpreting GPL over Markov Processes

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 M is a MP following from Definition 2.9 implies that $\Theta_M(\langle a\rangle\psi) = \Theta_M([a]\psi)$.

In the rest of this subsection we will show that pCTL* can be encoded in GPL. What makes the encoding possible are that:

- The logic GPL is a two-level logic, much like CTL* and pCTL*. Consequently, probabilistic quantifiers in pCTL* formulae can be translated to probabilistic quantifier of GPL formulae.
- The semantics of fuzzy formulae are sets of sequences, when the model is a Markov chain, and. thus, fuzzy formulae play the role of linear-time μ-calculus formulae. Given that alternation-free linear time modal μ-calculus is as expressive as linear time temporal logic (LTL) [Sti92], the LTL portion of pCTL* (i.e., the path formulae of pCTL*) can be embedded into fuzzy formulae.

This encoding contrasts with the encoding of CTL* into modal μ-calculus [EL86], where alternation is needed in the translation; the reason being that, unlike GPL, modal μ-calculus does not have path quantifiers.

3.2.2 pCTL*

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

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

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

The semantics of pCTL* formulas is given with respect to a MP $M = (S, \delta, P, I)$ via a relation $\models_M$ relating states in M to state formulas and paths (infinite state sequences) in M to path formulas. The interpretations of $\neg$ and $\vee$ are standard, and we omit them; what follows defines the meanings of the other operators.

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

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

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

We now have the following.

Theorem 3.3 *Let M be a MP, let s be a state in M, and let π be a path in M. Then:*

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

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 PLTS. 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 semantics of our fuzzy formulas 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 $Æ_{\geq p}$ and $Æ_{>p}$ is allowed. They then present three semantics for the logic that differ only in their interpretation of conjunction. Each interprets formulas as functions mapping states to numbers in $[0, 1]$; formally, given PLTS L, $[\![\psi]\!]_L : S \to [0, 1]$ represents the interpretation of formula ψ. What follows presents the relevant portions of these semantics.

$$\begin{array}{rcl} [\![\mathsf{tt}]\!]_L(s) & = & 1 \\ [\![\langle a \rangle \psi]\!]_L(s) & = & \displaystyle\sum_{s' \in \delta(s,a)} P(s, a, s') \cdot [\![\psi]\!]_L(s') \\ [\![\psi_1 \wedge \psi_2]\!]_L(s) = & = & f([\![\psi_1]\!]_L(s), [\![\psi_2]\!]_L(s)) \end{array}$$

The meanings of the other boolean and modal operators may be obtained using dualities (e.g. $[\![[a]\psi]\!]_L(s) = 1 - ([\![\langle a \rangle \neg\psi]\!])$, 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 not the case that $\mathtt{tt}$ 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; "real" probabilities are therefore not calculated.

GPL permits a similar interpretation to be attached to the mu-calculus, but in such a way that exact probabilities may be assigned to formulas. Consider the function $[\![\psi]\!]_L^{GPL}$ given by:

$$[\![\psi]\!]_L^{GPL}(s) = \mathsf{m}_s(\Theta_L(\psi)).$$

One can show that this interpretation preserves much of the semantics of Huth and Kwiatkowska; in particular, Lemmas 2.13 and 2.14 show that this definition attaches the same interpretations to the modalities. It is also the case that expected logical equivalences hold, and that this interpretation yields a probability with a precise, measure-theoretic interpretation. Finally, it should be easy to observe that our logic coincides with probabilistic bisimulation [LS91] – a property not true of Huth and Kwiatkowska's interpretation.

4 Model Checking

This section now describes a procedure for determining whether or not a given state in a finite-state PLTS 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 PLTS 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 formulas to be considered have no *unguarded* occurrences of bound variables. That is, in every sub-formula 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. In the remainder of this section we fix a specific PLTS $L = (S, \delta, P, I)$.

4.1 Computing the Measure of Fuzzy Formulas

This subsection sketches a procedure *modchk-fuzzy* whose task is to compute $\mathsf{m}_{s_0}(\Theta_L(\psi))$ for a given fuzzy formula ψ and a state s_0 of the PLTS. 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; one of the results will be $\mathsf{m}_{s_0}(\Theta_L(\psi))$.

The remainder of this subsection describes each of these steps in more detail. The appendix contains some discussion about their correctness, and thus here we only present intuitive explanations about why the constructions work.

A graph construction. The first step in *modchk-fuzzy* involves constructing a graph that describes the relationship between the quantity $\mathsf{m}_{s_0}(\Theta_L(\psi))$ that we wish to compute and quantities of the form $\mathsf{m}_s(\Theta_L(\psi'))$, where s is a derivative of s_0 and ψ' a formula derived appropriately from ψ. This graph will have vertices of the form (s, F), where $s \in S$ and F is a set of fuzzy formulas. The edges from (s, F) then provide "local" information regarding $\mathsf{m}_s(\Theta_L(\wedge F))$.

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

Definition 4.1 For a closed fuzzy formula ψ define the closure, written as $Cl(\psi)$, as the smallest set of formulae satisfying the following rules:

- $\psi \in Cl(\psi)$
- 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 sub-formulas.

The node set N in the graph is the set $S \times 2^{Cl(\psi)}$; that is,nodes have form (s, F), where $s \in S$ and $F \subseteq Cl(\psi)$. We further introduce the following classification on nodes.

- (s, F) is a *true node* if $F = \emptyset$ or if every element of F has form $[a]\psi'$ and for every such a, s is incapable of an a-transition.
- (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 s is incapable of an a-transition.
- (s, F) is an *and-node* if there exists a formula $\psi_1 \wedge \psi_2 \in F$.
- (s, F) is an *action-node* if every formula in F has form $\langle a \rangle \psi'$ or $[a]\psi'$.
- (s, F) is a *μ-node* if there exists a formula $\psi' \in F$ containing a top-level fixpoint sub-formula of form $\mu X.\psi''$; it is a *ν-node* otherwise.

Note that these categories overlap one another.

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 a true node or a false node,[2] then n is a sink node;
2. else if (s, F) contains state formulas then $((s, F), \epsilon^+, (s, F')) \in E$, where F' is F with all state formulas deleted;
3. else if (s, F) contains a fixpoint formula $\psi' = \sigma X.\psi''$ (where σ is μ or ν) then $((s, F), \epsilon^+, (s, F - \{\psi'\} \cup \{\psi''[\psi'/X]\})) \in E$;
4. else if $\psi = \psi_1 \wedge \psi_2 \in F$ then $((s, F), \epsilon^+, (s, F - \{\psi\} \cup \{\psi_1, \psi_2\})) \in E$;

[2]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.

5. else if (s, F) is not an and-node and $\psi = \psi_1 \vee \psi_2 \in F$ then $((s, F), \epsilon^+, (s, F - \{\psi\} \cup \{\psi_1\}) \in E$, $((s, F), \epsilon^+, (s, F - \{\psi\} \cup \{\psi_2\}) \in E$, and $((s, F), \epsilon^-, (s, F - \{\psi\} \cup \{\psi_1, \psi_2\})) \in E$;
6. else if (s, F) is an action node, let $F_a = \{\psi' \mid \langle a\rangle\psi' \in F \text{ or } [a]\psi' \in F\}$. Then for any $a \in Act$ with $F_a \neq \emptyset$ and $s' \in S$ such that $(s, a, s') \in \delta$, $((s, F), a, (s', F_a)) \in E$.

Intuitively, an edge $((s, F), \ell, (s', F'))$ indicates a "local relationship" between $\mathsf{m}_s(\Theta_L(\wedge F))$ and $\mathsf{m}_{s'}(\Theta_L(\wedge F'))$. To see this, first note that if (s, F) is a true node (false node) then $\mathsf{m}_s(\Theta_L(\wedge F)) = 1(0)$. Now suppose that (s, F) is an or-node to which case 5 applies. This means that $F = F' \cup \{\psi_1 \vee \psi_2\}$, 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 2.13 we may therefore conclude the following.

$$\mathsf{m}_s(\Theta_L(\wedge F)) = \mathsf{m}_s(\Theta_L(\wedge F' \wedge \psi_1)) + \mathsf{m}_s(\Theta_L(\wedge F' \wedge \psi_2)) - \mathsf{m}_s(\Theta_L(\wedge(F' \cup \{\psi_1, \psi_2\})))$$

This observation is encoded in the ϵ^+ and ϵ^- edges emanating from (s, F). Similar observations hold for the other nodes, with the exception of action nodes, which we discuss in more detail below.

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_n, for each node n in the graph and one equation containing this variable as its left-hand side. The right-hand side of the equation for X_n is generated as follows, based on the edges emanating from n.

1. If n is a true node then the equation for X_n is $X_n = 1$; if n is a false node, the equation for X_n is $X_n = 0$.
2. If there is an edge of the form (n, ϵ^+, n') then the equation for X_n is
$$X_n = \sum_{(n,\epsilon^+,n') \in E} X_{n'} - \sum_{((n,\epsilon^-,n') \in E} X_{n'}.$$
3. If $n = (s, F)$ is an action node, let $A_n = \{a \mid (n, a, n') \in E\}$. Then the equation for X_n is
$$X_n = \prod_{a \in A_n} \sum_{(n,a,(s',F')) \in E} (P(s, a, s') \cdot X_{(s',F')}).$$

Intuitively, these equations are intended to reflect relationships among the measures associated to each vertex. The right-hand side of in the equation associated to an action node reflects this intuition. A small example illustrates why. Suppose that action node (s, F) is such that $F = \{\langle a\rangle\psi_1, \langle b\rangle\psi_2\}$. Since this is not a false node, it follows that s has both a- and b-transitions. The question is, what is the measure of observations rooted at s and satisfying $\wedge F$? Each such observation would select one a-transition and one b-transition from s, with the target of the a-transition then being the root of an observation satisfying ψ_1 and similarly for the target of the b-transition. For a given combination of single a- and b-transitions with target states s_a and s_b, the measure of observations using these transitions and satisfying $\wedge F$ is $P(s, a, s_a) \cdot \mathsf{m}_{s_a}(\Theta_L(\psi_1)) \cdot P(s, b, s_b) \cdot \mathsf{m}_{s_b}(\Theta_L(\psi_2))$. Using simple symbol pushing, it is then easy to show that the total measure of observations emanating from s and satisfying $\wedge F$ is characterized by the right-hand side of the equation above.

We now have the following.

Lemma 4.2 *Let $\vec{E} = \{X_n = E_n\}$ be the equations generated above, and let $\vec{A}$ be the "vector" $\{X_n = \mathsf{m}_s(\Theta_L(\wedge F))\}$, where $n = (s, F)$. Then A is a solution to $\vec{E}$.*

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* does so as follows.

1. Compute the strongly connected components of the graph from which the equations are constructed and topologically sort them.
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 the variable.
3. Beginning at the end of the strongly connected component list, 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 values don't change (or fall within a tolerance ϵ that is a parameter to the algorithm).
   (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 have the following.

Lemma 4.3 *Let $s \in S$ and ψ be a fuzzy formula. Then the quantity calculated for $X_{(s,\{\psi\})}$ converges to $\mathsf{m}_s(\Theta_L(\psi))$.*

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 $\text{Æ}_{\geq p}\psi$ or $\text{Æ}_{>p}\psi$. In these latter two cases, *modchk-fuzzy* is called to calculate $\mathsf{m}_s(\Theta_L(\psi))$, 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 about complexity

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 PLTS is in fact a MP then the equation system generated is linear. In addition, results of [CY88] suggest that this linear system can be converted into one that has a unique solution. In this case, the equations can be solved exactly.

The non-linearity of the equations we consider, for model-checking PLTS, is a direct consequence of the program model (which allows different kinds of actions, i.e., when $|Act| > 1$) and our semantics (where observations are deterministic trees). Consequently, non-linearity in the measure equations is a fact that any solution technique, we adopt, will have to contend with. Furthermore, since there can be no direct technique for solving arbitrary polynomial equations (due to a result of Gauss) we will have to depend upon iterative techniques. A characteristic of iterative techniques, shared by our work, is that the complexity depends upon the precision of answers demanded. We have been investigating the use of symbol algebra tools, such as Maple, in implementing our model-checking procedure. and hope to report our experiences in the near future.

5 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 encode two different existing temporal logics, and we have also demonstrated that it may be used to rectify an infelicity in a third. A model-checking procedure for the logic was also presented.

As for future work, we believe that we can improve on the algorithm presented here by using results similar to those in [CY88] to transform our equation systems into ones having unique solutions. If this is the case, then we can use traditional solution techniques for nonlinear equations to compute measures in a numerically robust manner. We would also like to implement these algorithms. Another important issue for future work is that of applying our logic to more general transition systems (for example, the transition systems of [Seg95]) and establishing its relation to probabilistic automata[Paz71]. Such an extension would allow translation of pCTL* interpreted over probabilistic non-deterministic systems into our framework, much like the translation we have shown in this paper, and provide an efficient model-checking procedure for the same. 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].

The work being presented here also has applications to edge-profile-driven data flow-analysis [Ram96, BGS98], where the likelihood with which program properties hold is calculated; such calculation can then be used to perform profile-driven optimization [BL92]. Recent work [Ste91, Sch98] on reducing traditional data flow analysis problems to a model-checking problem can be extended to reduce profile driven DFA to probabilistic model-checking, and we propose to investigate this further.

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

References

[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.

[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.

[BGS98] Rastislav Bodik, Rajiv Gupta, and Mary Lou Soffa. Complete removal of redundant computations. *ACM SIGPLAN Notices*, 33(5):1–14, May 1998.

[BL92] Thomas Ball and James R. Larus. Optimally profiling and tracing progranis. In Ravi Sethi, editor, *Proceedings of the 19th Annual Symposium on Principles of Programming Languages*, pages 59–70, Albuquerque, NM, January 1992. ACM Press.

[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.

[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.

[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, Twelth 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, New Jersey, 1966.

[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.

[Paz71] Azaria Paz. *Introduction to Probablistic Automata*. Academic Press, New York, 1971.

[PSC+98] A. Philippou, O. Sokolksy, R. Cleaveland, I. Lee, and S. Smolka. Probabilistic resource failure in real-time process algebra. 1998. To appear in Proceedings of *CONCUR*.

[PZ93] Amir Pnueli and Lenore D. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, March 1993.

[Ram96] G. Ramalingam. Data flow frequency analysis. In *Proceedings of the ACM SIGPLAN '96 Conference on Programming Language Design and Implementation*, pages 267–277, Philadelphia, Pennsylvania, 21–24 May 1996.

[Sch98] David A. Schmidt. Data flow analysis is model checking of abstract interpretation. In *Conference Record of POPL '98: The 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 38–48, San Diego, California, 19–21 January 1998.

[Seg95] R. Segala. A compositional trace-based semantics for probabilistic automata. In *CONCUR95*, pages 324–338, 1995.

[SEJ93] A. P. Sistla, A. Emerson, and C. Jutla. Model checking in fragments of the mu-calculus. In *Proceedings of the International Conference on Computer Aided Verification*, volume Vol 697 of *LNCS*, pages 385–396. Springer-Verlag, June 1993.

[Ste91] Bernhard Steffen. Data flow analysis as model checking. *Lecture Notes in Computer Science*, 526:346–??, 1991.

[Sti92] C. Stirling. *Modal and Temporal Logics*, volume 2 of *Handbook of Logic in Computer Science*, pages 478–551. Oxford Science 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.

[vGSST90] 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.

A The product construction

In this appendix we will provide details of why the graph $G_{L.\psi,\hat{s}_0}$, when treated as a Büchi automaton, captures the set of d-trees in $\Theta_{L,s}(\psi)$. Before we proceed we note that Büchi automaton is at least as expressive as alternation-free μ-calculus [SEJ93]; this allows one to reduce the model-checking problem to checking emptiness of trees accepted by an appropriate Büchi automaton. In this section we will:

- Given an alternation-free fuzzy formula ψ, devoid of any Æ quantifiers, define a Büchi tree automaton (on infinite trees) $\mathcal{A}(\psi)$ which captures the models of ψ. We prove that the trees accepted by the automaton are precisely the models of ψ.
- Given a PLTS L, and a start state $\hat{s}_0$ we show how the graph $G_{L,\psi,\hat{s}_0}$ can be seen as tree automaton which specifies the d-trees that are common to L and the trees of $\mathcal{A}(\psi)$. This establishes the correctness of the graph construction.

We now proceed with the details:

Definition A.1 A tree automaton for a fuzzy formula ψ, over a set of propositions AP, which does not have any occurrences of Æ sub-formulae, is: $\mathcal{A}(\psi) = (Q, \delta_\epsilon \cup \delta_{act}, \mathcal{F}, Q_T, q_0, AP)$ where

- $Q = Q_{act} \cup Q_\epsilon \subseteq 2^{Cl(\psi)}$ is a set of states, where $q \in Q_{act}$ contains only modal formulae of the kind $\langle a\rangle\psi$ and $[a]\psi$. All other sets of formulae are in Q_ϵ.
- $\delta_\epsilon : Q \times \{\epsilon, \epsilon^+\} \times Q$, is the ϵ transition relation. An ϵ-transition function is one of the following:
  - $q \xrightarrow{\epsilon} q'$, provided q has a atomic propositions and its negations which are mutually consistent (i.e., both P and $\neg P$ are not in q). In this case $q' = q - (AP \cup \neg AP)$.
  - If $[a]\psi' \in q$, for some $a \in Act$ then $q \xrightarrow{\epsilon} q'$ where $q' = q - \{[a]\psi' \in q\}$.
  - $q \xrightarrow{\epsilon} q'$ provided q does not contain any atomic propositions or its negations, q either contains a $\psi_1 \wedge \psi_2$, $\mu X.\psi'$ or a $\nu X.\psi'$. If q contains $\psi_1 \wedge \psi_2$ then $q' = (q - \{\psi_1 \wedge \psi_2\}) \cup \{\psi_1, \psi_2\}$. If $\mu X.\psi'$ is in q then $q' = (q - \{\mu X.\psi'\}) \cup \{\psi'[\mu X.\psi'/X]$. Similarly, if $\nu X.\psi'$ is in q then $q' = (q - \{\nu X.\psi'\}) \cup \{\psi'[\nu X.\psi'/X]$.
  - $q \xrightarrow{\epsilon^+} q_1$ and $q \xrightarrow{\epsilon^+} q_2$ provided q does not contain any atomic propositions or negated propositions, and does not contain any formula with conjunction, μ or ν as the root operator. Furthermore, q contains a disjunction of the form $\psi_1 \vee \psi_2$. Under such a situation $q_1 = (q - \{\psi_1 \vee \psi_2\}) \cup \{\psi_1\}$ and $q_2 = (q - \{\psi_1 \vee \psi_2\}) \cup \{\psi_2\}$.
- $\delta_{act} \subseteq Q_{act} \times Act \times Q$ is the action transition function. δ_{act} is defined as follows:

$$\delta_{act}(q, a) = \{\psi' \mid [a]\psi' \in q \vee \langle a\rangle\psi' \in q\}$$

- $\mathcal{F}$ – the set of final states – is the set of all states q such that all occurrences of a μ sub-formula in q is enclosed in some ν-formula.
- $q_0 = \{\psi\}$ is the start state of the system.

Notation: ρ ranges over 2^{AP}.

A tree T over atomic propositions AP and actions Act is a set of sequences over $2^{AP}(Act \cdot 2^{AP})^*$ such that it is prefix closed and is deterministic. We will follow earlier conventions by writing a sequence as $\rho_0 \xrightarrow{a_0} \rho_1 \xrightarrow{a_1} \ldots \rho_n$. The root of a tree T is the unique ρ_0 such that all sequences in T are of the form $\rho_0 \xrightarrow{a_0} \cdots$ for some a_0.

A run of tree automaton $\mathcal{A}(\psi)$ on a tree T is a tree $r(T, \mathcal{A}(\psi))$ written as a set of deterministic, prefix-closed sequences over $(Q \times 2^{AP})((Act \cup \{\epsilon\}) \cdot (Q \times 2^{AP}))^*$ which satisfies certain constraints. To aid in the discussion of these constraints we now define the projection of a sequence $\zeta \in r(T, \mathcal{A}(\psi))$ on T, written $\zeta \downarrow T$ as follows (where $a_i \in Act$ and $\hat{a}_i \in Act \cup \{\epsilon\}$):

$$
\begin{array}{l}
(q_0, \rho_0) \downarrow T = \rho_0 \\
\zeta' \xrightarrow{\hat{a}_{n-1}} (q_n, \rho_n) \xrightarrow{\epsilon} (q_{n+1}, \rho_{n+1}) \downarrow T = \zeta' \xrightarrow{\hat{a}_{n-1}} (q_n, \rho_n) \downarrow T \\
\zeta' \xrightarrow{\hat{a}_{n-1}} (q_n, \rho_n) \xrightarrow{a_n} (q_{n+1}, \rho_{n+1}) \downarrow T = (\zeta' \xrightarrow{\hat{a}_{n-1}} (q_n, \rho_n) \downarrow T) \xrightarrow{a_n} \rho_{n+1}
\end{array}
$$

The constraints that a run $r(T, \mathcal{A}(\psi))$ should satisfy are as follows:

- $\mathsf{root}(r(T, \mathcal{A}(\psi)))$ is $(q_0, \mathsf{root}(T))$, where q_0 is the start state of $\mathcal{A}(\psi)$..
- If ζ is a sequence in $r(T, \mathcal{A}(\psi))$ such that $\mathsf{lst}(\zeta) = (\emptyset, \rho)$ and $\zeta \downarrow T = \eta$, then for all $a \in Act$ and for all $\rho' \in 2^{AP}$ such that $\eta \xrightarrow{a} \rho' \in T$ we have $\zeta \xrightarrow{a} (\emptyset, \rho') \in r(T, \mathcal{A}(\psi))$. This condition states that if a node in a tree has no requirements imposed by the automaton then none of the successors in that subtree have any requirements either.
- If ζ is a sequence in $r(T, \mathcal{A}(\psi))$ such that $\mathsf{lst}(\zeta) = (q, \rho)$, $(AP \cup \neg AP) \cap q \neq \emptyset$ and $\delta_\epsilon(q, \sigma) = q'$ then $(q - q') \cap AP \subseteq \rho$, $\{\alpha | \neg\alpha \in (q - q')\} \cap \rho = \emptyset$ and $\zeta \xrightarrow{\epsilon} (q', \rho) \in r(T, \mathcal{A}(\psi))$. This requirements checks the constraint that a set of atomic propositions be true (or not true) at a node in a tree.
- If ζ is a sequence in $r(T, \mathcal{A}(\psi))$ such that $\mathsf{lst}(\zeta) = (q, \rho)$, $q \in Q_\epsilon$, $q' \in \delta_\epsilon(q, \epsilon)$ then $\zeta' = \zeta \xrightarrow{\epsilon} (q', \rho)$ is in $r(T, \mathcal{A}(\psi))$. Note that the sequence ζ has exactly one-successor.
- Let ζ be a sequence in $r(T, \mathcal{A}(\psi))$ such that $\mathsf{lst}(\zeta) = (q, \rho)$, and $\eta = \zeta \downarrow T$. If there is an $a \in Act$ such that $[a]\psi' \in q$ and there is no $\eta' \in T$ such that $\eta \xrightarrow{a} \eta'$ then $\zeta \xrightarrow{\epsilon} (q - \{[a]\psi' \in q\}, \rho)$ is a sequence in $r(T, \mathcal{A}(\psi))$.
- Let ζ be a sequence in $r(T, \mathcal{A}(\psi))$ such that $\mathsf{lst}(\zeta) = (q, \rho)$, $q \in Q_{act}$, $\eta = \zeta \downarrow T$. If for all $a \in Act$ such that $\delta_{act}(q, a)$ is non-empty there is a ρ_a such that $\eta \xrightarrow{a} \in T$ then the run should satisfy the constraint:

  for each $a \in Act$ such that $\delta(q, a) = q_a$ is non-empty there is a ρ_a such that $\eta' = \eta \xrightarrow{a} \rho_a$ in T and $\zeta' = \zeta \xrightarrow{a} (q_a, \rho_a) \in r(T, \mathcal{A}(\psi)$. Furthermore, $\zeta' \downarrow T = \eta'$.

A run $r(T, \mathcal{A}(\psi))$ is considered acceptable provided all infinite paths ζ (i.e., the limit of sequences in $r(T, \mathcal{A}(\psi)))$ satisfy the following constraint: (a) an infinite tail of ζ is of the form $(\emptyset, \rho)$, for various ρ, or (b) some state $q_f \in \mathcal{F}$ is visited infinitely often in ζ. More formally, for every limit of sequences in $r(T, \mathcal{A}(\psi))$ states of the form $(\emptyset, \rho)$ or (q_f, ρ) appear infinitely often in the limiting sequence.
Notation: Consider a d-tree T of PLTS L rooted at s. Define $\hat{T}$ to be homomorphic image of T under the interpretation mapping $I : \mathcal{S} \to 2^{AP}$. Such a homomorphic image is necessary as trees in L are labeled by the states of L, where as we are interested in the atomic propositions that are true in those states.

Theorem A.2 *Let T be a d-tree of PLTS L with root s. For every fuzzy formula ψ devoid of any Æ quantifiers we have $T \in \Theta_{L,s}(\psi)$ iff $\hat{T}$ is accepted by the automaton $\mathcal{A}(\psi)$.*

Proof (sketch)
$\Rightarrow$The result in [Cle90] provides a set of syntactic proof rules for checking that a finite state label transition system satisfies a μ-calculus formula and proves the soundness and completeness of those rules. A valid proof in such a system is a finite acyclic graph, with the root node is labeled by a sequent $s_0 \vdash \psi_0$ (saying that s_0 satisfies the property ψ_0), and the leaf nodes are labeled by either true or a sequent of the form $s \vdash \nu X.\psi$ which happens to appear on some other node on the path from the root node to that leaf (there by creating a implicit cycle in the reasoning process). Furthermore, another strict condition imposed is that no valid proof can contain a path from a node labeled by a μ-formula to another node labeled by the same μ-formula. Consequently, the presence of the same ν-property on two nodes from the root to a leaf engenders an implicit cycle. Based on such cycles a proof tree could be unrolled to obtain a (potentially) infinite tree. However, this tree could contain finite paths

that end in sequents of the form $s \vdash true$ or $s \vdash [a]\psi$ when there is no a-successor of s in the LTS. Such leaves (say $s \vdash true$) can be can be extended by attaching any tree rooted at s obtained by unrolling the LTS, and by labeling all nodes with the sequent of the form $s' \vdash true$. Such a transformation does not change the validity of the proof. More importantly, the resulting proof tree can be thought upon as a run of the automaton $\mathcal{A}(\psi)$.

The only difference between the proof rules and the constraints of a run in our system are (a) the sequents in a proof are of the form $s \vdash \psi$ and in a run the sequents (or what is equivalent to a sequent) is a pair of set of formulae and a state, and (b) the proof obligation for $s \vdash \psi_1 \wedge \psi_2$ in [Cle90] leads to branching in the proof tree where as in a run $\psi_1 \wedge \psi_2$ is broken down to form a set of simpler formulae ($\{\psi_1, \psi_2\}$). Thus a run of a tree s-rooted d-tree T, or equivalently $\hat{T}$, can be established by starting from an unrolled proof tree and merging the successors of sequents containing conjunctions.
$\Leftarrow$: Assume that there is a run for $\hat{\boldsymbol{T}}$ on $\boldsymbol{\mathcal{A}(\psi)}$, say tree $\boldsymbol{t}$. Consider a family of finite trees $\boldsymbol{t_i}$, where each $\boldsymbol{t_i}$ is prefix of the tree $\boldsymbol{t}$ containing nodes down to level $\boldsymbol{i}$. Transform each $\boldsymbol{t_i}$ such that the recursion operators $\boldsymbol{\nu}$ is replaced by approximants $\boldsymbol{\nu^j}$ such the occurrence of a formula nearest to the leaf has $\boldsymbol{\nu^0}$ and the ones above it have increasing greater index. We now claim that every formula in the state $\boldsymbol{q}$ of $\boldsymbol{(q, \rho)}$ is true for the subtree of $\boldsymbol{t}$ (corresponding to $\boldsymbol{T}$) that emanates from that state. The claim follows from the fact that the constraints imposed by what a good "run" is are sound logical implications.

Assume a $\boldsymbol{\nu}$-formula appears infinitely often on an acceptable run. Now if we track that node across the entire family of trees $\boldsymbol{t_i}$, it will be labeled by $\boldsymbol{\nu^j}$ for increasing $\boldsymbol{j}$s, thus establishing that all the approximants of the $\boldsymbol{\nu}$ formula are true for the tree emanating from that node. This establishes that the entire tree $\boldsymbol{T}$ is in $\boldsymbol{\Theta_{L,s}(\psi)}$. ■

Construction of the graph Given that $\mathcal{A}(\psi)$ captures the set of trees that are models of ψ and given that we are interested in s-rooted d-tress of the PLTS L, we can think of $G_{L,\psi,\hat{s}_0}$ as the cross product of the automaton $\mathcal{A}(\psi)$ and PLTS L, resulting in a tree automaton. Note the similarity between the construction of $\mathcal{A}(\psi)$ and $G_{L,\psi,\hat{s}_0}$. Note that there is only one difference between the two constructions; $G_{L,\psi,\hat{s}_0}$ has three successors for every or-node while $\mathcal{A}(\psi)$ has only two successors. However, we can recover $G_{L,\psi,\hat{s}_0}$ by starting with product of L and $\mathcal{A}(\psi)$, and the product of L and the formulae in the ϵ^- successors of each or-node in the first graph. These graphs can be then merged together where common nodes are identified. Clearly, this product construction and merging needs to be carried out recursively so that all or-nodes are treated. However, the recursive construction is guaranteed to terminate by finiteness of Fisher-Ladner closure of any μ-calculus formula. By the results of our previous section we can treat $G_{L,\psi,\hat{s}_0}$ as a tree automaton with the final states being those states (s, F) where F contains formulae where all μ-formula are enclosed in a ν-formula. We consequently have:

Theorem A.3 *The set of trees accepted by the tree automaton $G_{L,\psi,\hat{s}_0}$ is $\Theta_{L,\hat{s}_0}(\psi)$.*