

The Satisfiability and Validity Problems for Probabilistic Computational Tree Logic are Highly Undecidable

Miroslav Chodil 

Faculty of Informatics, Masaryk University, Brno, Czechia

Antonín Kučera 

Faculty of Informatics, Masaryk University, Brno, Czechia

Abstract

The Probabilistic Computational Tree Logic (PCTL) is the main specification formalism for discrete probabilistic systems modeled by Markov chains. Despite serious research attempts, the decidability of PCTL satisfiability and validity problems remained unresolved for 30 years. We show that both problems are *highly undecidable*, i.e., beyond the arithmetical hierarchy. Consequently, there is no sound and complete deductive system for PCTL.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases Satisfiability, temporal logics, probabilistic CTL

Funding The presented work received funding from the European Union’s Horizon Europe program under the Grant Agreement No. 101087529.

1 Introduction

Algorithms for checking the satisfiability/validity of formulae in a given logic have received significant attention from the very beginning of computer science. The satisfiability/validity problems are closely related to each other because for most logics we have that φ is valid iff $\neg\varphi$ is not satisfiable. The decidability of the validity problem for first-order logic, also known as the *Entscheidungsproblem*, was posed by Hilbert and Ackermann in 1928 and answered negatively by Church [8] and Turing [23] in 1936. Later, Trakhtenbrot [22] demonstrated that the *finite* satisfiability problem for first-order logic is also undecidable. Consequently, the *finite validity* problem is not even semi-decidable, and hence there is no sound and complete deductive system proving all finite valid first-order formulae. For propositional logic, the satisfiability and validity are the canonical **NP** and **coNP** complete problems [9].

The satisfiability and validity problems have also been intensively studied for *temporal logics* such as LTL, CTL, CTL*, or the modal μ -calculus, and their computational complexity has been fully classified (see, e.g., [10, 21]). Temporal logics are interpreted over *transition systems*, i.e., directed graphs where the vertices correspond to the states of an abstract non-deterministic program, and the edges model the atomic computational steps. Even for the most expressive logic of the modal μ -calculus [14], the satisfiability/validity problems are decidable and **EXPTIME**-complete, and every satisfiable formula φ has a model whose size is at most exponential in $|\varphi|$ (this is known as “the small model property” [15]). Furthermore, there is a sound and complete deductive system for the modal μ -calculus [24].

Probabilistic temporal logics, such as PCTL or PCTL* [11] are interpreted over discrete-time Markov chains modelling *probabilistic programs*. Despite numerous research attempts resulting in partial positive results (see Related work), the decidability of the satisfiability/validity problems for PCTL has remained unresolved for more than 30 years. In this paper, we prove that both problems are *highly undecidable* (i.e., beyond the arithmetical hierarchy), even for a simple PCTL fragment where the path connectives are restricted to **F**

and **G**. Consequently, there is no sound and complete deductive system for PCTL.

Related work. Unlike non-probabilistic temporal logics, PCTL does not have the small model property. In fact, one can easily construct satisfiable PCTL formulae without *any* finite model (see, e.g., [3]). Hence, the PCTL satisfiability/validity problems have also been studied in their finitary variants.

The first positive decidability results have been obtained for the *qualitative fragment* of PCTL, where the range of admissible probability constraints is restricted to $=0$, >0 , $=1$, and <1 (see Section 2 for details). The satisfiability/validity for qualitative PCTL is **EXPTIME**-complete, and the same holds for finite satisfiability/validity [19, 13, 16]. Furthermore, a finite description of a model for a satisfiable qualitative PCTL formula is constructible in exponential time [3]. The underlying proof techniques are similar to the ones used for non-probabilistic temporal logics.

The finite satisfiability is decidable also for certain *quantitative* PCTL fragments with general probability constraints [17, 5, 7]. The underlying arguments are mostly based on establishing an upper bound on the size of a model for a satisfiable formula of the considered fragment (the existence of a model can then be expressed in the existential fragment of first-order theory of the reals, which is known to be in **PSPACE** [4]). More concretely, in [5] it is shown that every formula φ of the *bounded fragment* of PCTL, where the validity of φ in a state s depends only on a bounded prefix of a run initiated in s , has a bounded-size tree model. In [17], several PCTL fragments based on **F** and **G** operators are studied. For each of these fragments, it is shown that every *finite* satisfiable formula has a bounded-size model where every non-bottom SCC is a singleton. For some of these fragments, it is shown that every satisfiable formula has a finite model, which yields the decidability of the (general) satisfiability problem. In [7], more abstract decidability results about finite PCTL satisfiability based on isolating the progress achieved along a chain of visited SCCs are presented. A variant of the bounded satisfiability problem, where transition probabilities are restricted to $\{\frac{1}{2}, 1\}$, is proven **NP**-complete in [1]. A recent result of [6] says that the *finite* satisfiability problem for unrestricted PCTL is *undecidable*. For a given Minsky machine, a PCTL formula ψ is constructed so that every reachable configuration of the machine corresponds to a special state of a model. Hence, ψ is *always* satisfiable, and ψ has a *finite* model iff the set of reachable configurations of the machine is finite. Hence, the construction does not imply the undecidability of general PCTL satisfiability, and some of the crucial tools used in [6] apply only to finite Markov chains (see Section 4 for more comments).

Our contribution. We show that the satisfiability problem for PCTL is hard for the Σ_1^1 level of the analytical hierarchy. Consequently, the PCTL validity problem is Π_1^1 -hard. These results hold even for the PCTL fragment where only the path connectives **F**, **G** are allowed. The proof is based on encoding the existence of a recurrent computation of a given non-deterministic two-counter machine \mathcal{M} by an effectively constructible PCTL formula $\varphi_{\mathcal{M}}$. The overall structure of $\varphi_{\mathcal{M}}$ and the main underlying concepts are introduced in Section 3. In particular, in Section 3.2, we identify a fundamental obstacle that needs to be overcome when constructing $\varphi_{\mathcal{M}}$. This is achieved by utilizing some concepts of convex geometry, and the solution is presented in Section 4. The construction of $\varphi_{\mathcal{M}}$ is then described in more detail in Section 5. Interestingly, our results imply that the *finite* satisfiability of the **F**, **G**-fragment is also undecidable. This is a non-trivial refinement of the result achieved in [6], where the finite satisfiability is shown undecidable for PCTL formulae containing the **F**, **G**, **X**, and **F^k** path connectives.

The main *technical* contributions are the following. Non-negative integer counter values are represented by *pairs* of probabilities. More concretely, we fix a suitable $\mathbf{z} \in (0, 1)^2$

representing zero, and then design a suitable function $Inc : (0, 1)^2 \rightarrow (0, 1)^2$ such that every $n \geq 0$ is represented by the pair $Inc^n(\mathbf{z})$, where Inc^n denotes the n -fold composition of Inc . The function Inc is a modification of the function σ designed in [6]. Then, we prove that every (possibly *infinite*) $T \subseteq (0, 1)^2$ satisfying a certain closure property must be contained in the convex polytope determined by the set of points $\{Inc^n(\mathbf{z}) \mid n \geq 0\}$ (see Theorem 5). This is the crucial (novel) ingredient for overcoming the obstacles identified in Section 3.2. The second major technical contribution is a (novel) technique for representing the points $Inc^n(\mathbf{z})$ by pairs of PCTL formulae of the **F, G**-fragment.

The presented construction is non-trivial and technically demanding. We believe that, at least to some extent, this is caused by the inherent difficulty of the considered problem. We did our best to provide a readable sketch of the main steps in Section 5, where we also explain the purpose and use of Theorem 5. Full technical details are in Appendix.

2 Preliminaries

The sets of non-negative integers, rational numbers, and real numbers are denoted by \mathbb{N} , \mathbb{Q} , and \mathbb{R} , respectively. We use $\mathbf{u}, \mathbf{v}, \mathbf{z}, \dots$ to denote the elements of $\mathbb{R} \times \mathbb{R}$. The first and the second components of \mathbf{u} are denoted by \mathbf{u}_1 and \mathbf{u}_2 . For a function $f : A \rightarrow A$, we use f^n to denote the n -fold composition $f \circ \dots \circ f$.

2.1 The Logic PCTL

The logic PCTL [11] is obtained from the standard CTL (Computational Tree Logic [10]) by replacing the existential and universal path quantifiers with the probabilistic operator $P(\Phi) \bowtie r$. PCTL formulae are interpreted over Markov chains where every state s is assigned a subset $v(s) \subseteq AP$ of propositions valid in s .

► **Definition 1** (PCTL). *Let AP be a set of atomic propositions. The syntax of PCTL state and path formulae is defined by the following abstract syntax equations:*

$$\begin{aligned} \varphi &::= a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid P(\Phi) \bowtie r \\ \Phi &::= \mathbf{X}\varphi \mid \varphi_1 \mathbf{U} \varphi_2 \mid \varphi_1 \mathbf{U}^k \varphi_2 \end{aligned}$$

Here, $a \in AP$, $\bowtie \in \{\geq, >, \leq, <, =, \neq\}$, $r \in [0, 1]$ is a rational constant, and $k \in \mathbb{N}$.

A *Markov chain* is a triple $M = (S, Prob, v)$, where S is a finite or countably infinite set of *states*, $Prob : S \times S \rightarrow [0, 1]$ is a probability matrix, and $v : S \rightarrow 2^{AP}$ is a *valuation*. For $s, t \in S$, we say that t is an *immediate successor* of s if $Prob(s, t) > 0$. A *path* in M is a finite sequence $w = s_0, \dots, s_n$ of states where $n \geq 0$ and $Prob(s_i, s_{i+1}) > 0$ for all $i < n$. We say that t is *reachable* from s if there is a path from s to t . A *run* in M is an infinite sequence $\pi = s_0, s_1, \dots$ of states such that every finite prefix of π is a path in M . We also use $\pi(i)$ to denote the state s_i of π .

For every path $w = s_0, \dots, s_n$, let $Run(w)$ be the set of all runs starting with w , and let $\mathbb{P}(Run(w)) = \prod_{i=0}^{n-1} Prob(s_i, s_{i+1})$. To every state s , we associate the probability space $(Run(s), \mathcal{F}_s, \mathbb{P}_s)$, where \mathcal{F}_s is the σ -field generated by all $Run(w)$ where w starts in s , and \mathbb{P}_s is the unique probability measure obtained by extending \mathbb{P} in the standard way (see, e.g., [2]). The *validity* of a PCTL state/path formula for a given state/run of M is defined inductively

as follows:

$$\begin{aligned}
s \models a & \quad \text{iff} \quad a \in v(s), \\
s \models \neg \varphi & \quad \text{iff} \quad s \not\models \varphi, \\
s \models \varphi_1 \wedge \varphi_2 & \quad \text{iff} \quad s \models \varphi_1 \text{ and } s \models \varphi_2, \\
s \models P(\Phi) \bowtie r & \quad \text{iff} \quad \mathbb{P}_s(\{\pi \in \text{Run}(s) \mid \pi \models \Phi\}) \bowtie r, \\
\pi \models \mathbf{X} \varphi & \quad \text{iff} \quad \pi(1) \models \varphi \text{ for some } i \in \mathbb{N}, \\
\pi \models \varphi_1 \mathbf{U} \varphi_2 & \quad \text{iff} \quad \text{there is } j \geq 0 \text{ such that } \pi(j) \models \varphi_2 \text{ and } \pi(i) \models \varphi_1 \text{ for all } 0 \leq i < j, \\
\pi \models \varphi_1 \mathbf{U}^k \varphi_2 & \quad \text{iff} \quad \text{there is } 0 \leq j \leq k \text{ such that } \pi(j) \models \varphi_2 \text{ and } \pi(i) \models \varphi_1 \text{ for all } 0 \leq i < j.
\end{aligned}$$

We say that M is a *model* of φ if $s \models \varphi$ for some state s of M . The *PCTL satisfiability problem* is the question of whether a given PCTL formula has a model.

The formulae *true*, *false* and the other Boolean connectives are defined using \neg and \wedge in the standard way. In the following, we abbreviate a formula of the form $P(\Phi) \bowtie r$ by omitting P and adjoining the probability constraint directly to the topmost path operator of Φ . For example, we write $\mathbf{X}_{=1} \varphi$ instead of $P(\mathbf{X} \varphi) = 1$. We also use $\mathbf{F}_{\bowtie r} \varphi$ and $\mathbf{F}_{\bowtie r}^k \varphi$ to abbreviate the formulae $\text{true} \mathbf{U}_{\bowtie r} \varphi$ and $\text{true} \mathbf{U}_{\bowtie r}^k \varphi$, respectively. Furthermore, $\mathbf{G}_{\bowtie r} \varphi$ abbreviates $\mathbf{F}_{\bowtie r} \neg \varphi$. Hence, $s \models \mathbf{G}_{\bowtie r} \varphi$ iff $\mathbb{P}_s(\{\pi \in \text{Run}(s) \mid \pi(i) \models \varphi \text{ for all } i \geq 0\}) \bowtie r$.

2.2 Non-Deterministic k -Counter Machines

Our results are obtained by constructing a PCTL formula encoding the existence of a recurrent computation in a given non-deterministic two-counter machine. However, the standard Minsky machines [20] are not particularly convenient for this purpose. More concretely, since every instruction modifies only *one* of the counters, a faithful simulation requires a mechanism for “copying” the other counter value to the next configuration. In other words, we need to implement not only the test-for-zero, increment, and decrement instructions, but also the ‘no-change’ operation.

To avoid the need for implementing the ‘no-change’ operation, we introduce a special variant of non-deterministic counter machines where every instruction performs an operation on *all* counters simultaneously. Let $d \geq 1$. A non-deterministic d -counter machine \mathcal{M} is a finite sequence $\text{Ins}_1, \dots, \text{Ins}_m$ of instructions operating over non-negative counters C_1, \dots, C_d , where every Ins_ℓ takes the form

$$\langle C_k=0 ? Z : P \rangle : \text{update}_1, \dots, \text{update}_d \quad (1)$$

where $k \in \{1, \dots, d\}$, $Z, P \subseteq \{1, \dots, m\}$ are non-empty subsets of target instruction indexes, and $\text{update}_1, \dots, \text{update}_d \in \{\text{dec}, \text{inc}\}$ are counter updates. \mathcal{M} is *deterministic* if Z and P are singletons in all Ins_ℓ . We use C_k^ℓ , Z^ℓ , P^ℓ , and $\text{update}_1^\ell, \dots, \text{update}_d^\ell$ to denote the respective elements of Ins_ℓ .

The instruction (1) is executed as follows. First, the counter C_k is tested for zero, and a successor label ℓ is chosen non-deterministically from either Z or P , depending on whether the test succeeds or not, respectively. Then, $\text{update}_1, \dots, \text{update}_d$ simultaneously update the current values of C_1, \dots, C_d . Here, *inc* increments the counter by one, and *dec* either decrements the counter by one or leaves the counter unchanged, depending on whether its current value is positive or zero, respectively. After that, the computation continues with Ins_ℓ . Note that the zero test is used *only* to determine the next instruction label.

More precisely, a *configuration* of \mathcal{M} is a tuple of the form (ℓ, c_1, \dots, c_d) where $\ell \in \{1, \dots, m\}$ is the current instruction index, and $c_1, \dots, c_d \in \mathbb{N}$ are the current values of C_1, \dots, C_d . A configuration $(\ell', c'_1, \dots, c'_d)$ is a *successor* of (ℓ, c_1, \dots, c_d) ,

written $(\ell, c_1, \dots, c_d) \mapsto (\ell', c'_1, \dots, c'_d)$, if the following conditions are satisfied, where $Ins_\ell \equiv \langle C_k=0? Z : P \rangle : update_1, \dots, update_d$.

- If $c_k = 0$ then $\ell' \in Z$; otherwise $\ell' \in P$.
- For all $j \in \{1, \dots, d\}$ we have that c'_j is equal to either c_j+1 or $\max\{0, c_j-1\}$, depending on whether $update_j$ is *inc* or *dec*, respectively.

A *computation* of \mathcal{M} is an infinite sequence $conf_0, conf_1, \dots$ of configurations such that $conf_0 = (1, 0, \dots, 0)$ and $conf_i \mapsto conf_{i+1}$ for all $i \in \mathbb{N}$. A computation is *bounded* if it contains only finitely many pairwise different configurations, and *τ -recurrent*, where $\tau \subseteq \{1, \dots, m\}$, if it contains infinitely many configurations of the form (i, c_1, \dots, c_d) where $i \in \tau$.

In the next proposition, Σ_1^0 and Σ_1^1 denote the corresponding levels in the arithmetical and the analytical hierarchies, respectively. A proof is obtained by simulating the standard Minsky machines, see Appendix.

► **Proposition 2.** *The problem of whether a given non-deterministic two-counter machine \mathcal{M} has a τ -recurrent computation (for a given τ) is Σ_1^1 -hard. Furthermore, the problem of whether a given deterministic two-counter machine \mathcal{M} has a bounded computation is Σ_1^0 -hard.*

3 Simulating Non-Deterministic Two-Counter Machines by PCTL

In this section, we give a high-level description of our undecidability proof, together with some preliminary intuition about the underlying techniques. The presented ideas are elaborated in subsequent sections.

Let \mathcal{M} be a non-deterministic two-counter machine with m instructions. Our aim is to construct a PCTL formula $\varphi_{\mathcal{M}}$ such that

- every model of $\varphi_{\mathcal{M}}$ contains a run representing a recurrent computation of \mathcal{M} ;
- for every recurrent computation of \mathcal{M} there exists a model of $\varphi_{\mathcal{M}}$ representing the computation.

This entails the Σ_1^1 hardness of PCTL satisfiability.

In the following sections, we indicate how to construct $\varphi_{\mathcal{M}}$ for a given non-deterministic *one-counter* machine \mathcal{M} . The restriction to one counter leads to simpler notation without omitting any crucial ideas. The extension to non-deterministic *two-counter* machines is relatively simple (see Section 6). For the rest of this section, we fix a non-deterministic one-counter machine \mathcal{M} with m instructions.

3.1 Representing the configurations of \mathcal{M}

The first step towards constructing the formula $\varphi_{\mathcal{M}}$ is finding a way of representing configurations of \mathcal{M} by “PCTL-friendly” properties of states in Markov chains.

More concretely, a state s representing a configuration (ℓ, c) must encode the index ℓ and the non-negative integer c . The ℓ is encoded by a dedicated atomic proposition at_ℓ valid in s (for technical reasons, at_ℓ is actually a disjunction of several propositions). The way of encoding c is more elaborate. Intuitively, the only unbounded information associated to s that is “visible” to PCTL is the probability of satisfying certain path formulae in s . The value of c is encoded by a *pair* of probabilities $\gamma[s] \in (0, 1)^2$ of satisfying suitable path formulae Φ_s and Ψ_s in s . As we shall see, the formulae Φ_s and Ψ_s depend on the state s , but the only information about s used to determine Φ_s and Ψ_s is the (in)validity of certain atomic propositions in s . Consequently, there are only *finitely many* PCTL formulae used to encode

the counter values. The reason why c cannot be represented just by a *single* probability of satisfying some appropriate path formula is indicated in Section 3.2.

It remains to explain which pair of probabilities encodes a given integer. Zero is encoded by a fixed (suitably chosen) pair \mathbf{z} , and if n is encoded by \mathbf{v} , then $n + 1$ is encoded by $\text{Inc}(\mathbf{v})$, where $\text{Inc} : (0, 1)^2 \rightarrow (0, 1)^2$ is a suitable function defined in Section 4.

To sum up, a state s encodes a configuration (ℓ, c) iff $s \models \text{at}_\ell$ and $\gamma[s] = \text{Inc}^c(\mathbf{z})$ (recall that Inc^c denotes the c -fold composition of Inc).

3.2 Simulating a computational step of \mathcal{M}

The crucial part of our construction is the simulation of one computational step of \mathcal{M} . Let $\ell \in \{1, \dots, m\}$ where $\text{Ins}_\ell \equiv \langle C=0? Z : P \rangle : \text{update}$. We show that for every $\ell' \in Z \cup P$, there exists a formula $\text{Step}_{\ell, \ell'}$ satisfying the following property:

► **Property 1.** *For every computational step of the form $(\ell, c) \mapsto (\ell', c')$ and every state s representing (ℓ, c) we have the following: If $s \models \text{Step}_{\ell, \ell'}$, then*

- (i) *there exist $\pi \in \text{Run}(s)$ and $j \geq 1$ such that $\pi(j)$ represents the configuration (ℓ', c') ;*
- (ii) *for every $\pi \in \text{Run}(s)$ and every $j \geq 1$ such that $\pi(j)$ represents a configuration of \mathcal{M} , there exists $1 \leq j' \leq j$ such that $\pi(j')$ represents the configuration (ℓ', c') .*

The construction of $\text{Step}_{\ell, \ell'}$ is perhaps the most tricky part of our proof. At first glance, the above features (i) and (ii) seem *unachievable* because of the following fundamental obstacle.

Suppose that s represents (ℓ, c) and $s \models \text{Step}_{\ell, \ell'}$. For simplicity, let us assume that the underlying Markov chain of s has been unfolded into an infinite tree. According to (i), there exist $\pi \in \text{Run}(s)$ and $j \geq 1$ such that $\pi(j)$ represents the configuration (ℓ', c') . For brevity, let us denote the states $\pi(j-1)$ and $\pi(j)$ by u and t , respectively. Since t represents (ℓ', c') , we have that $\gamma[t] = \text{Inc}^{c'}(\mathbf{z})$ (see Section 3.1). Due to Property 1, (i) and (ii) must hold in an arbitrary modification \mathcal{M}' of \mathcal{M} such that s still represents (ℓ, c) and satisfies $\text{Step}_{\ell, \ell'}$ in \mathcal{M}' . Now consider a modification where the immediate successor t of u is replaced with t_1, \dots, t_n so that

- (a) every t_i satisfies the same set of atomic propositions as t ,
- (b) $\text{Prob}(u, t) = \sum_{i=1}^n \text{Prob}(u, t_i)$,
- (c) $\text{Prob}(u, t) \cdot \mathbb{P}(\Phi, t) = \sum_{i=1}^n \text{Prob}(u, t_i) \cdot \mathbb{P}(\Phi, t_i)$ for every path subformula Φ of $\text{Step}_{\ell, \ell'}$.

Here, $\mathbb{P}(\Phi, q)$ is the probability of all runs initiated in q satisfying Φ .

Since this modification does not change the value of $\mathbb{P}(\Phi, s)$ for any path subformula Φ of $\text{Step}_{\ell, \ell'}$, we have that $s \models \text{Step}_{\ell, \ell'}$ after the modification. Note that (c) implies $\text{Prob}(u, t) \cdot \gamma[t] = \sum_{i=1}^n \text{Prob}(u, t_i) \cdot \gamma[t_i]$. However, if we choose t_1, \dots, t_n so that every $\gamma[t_i]$ is *different* from $\gamma[t]$, we may spoil both (i) and (ii).

Since spoiling the envisioned functionality of $\text{Step}_{\ell, \ell'}$ is so simple, one is even tempted to conclude the *non-existence* of $\text{Step}_{\ell, \ell'}$. Indeed, no suitable $\text{Step}_{\ell, \ell'}$ can exist without some extra restriction preventing the use of *arbitrary* t_1, \dots, t_n satisfying (a)–(c). We implement such a restriction with the help of basic convex geometry. First, observe that the condition (c) implies that $\gamma[t]$ is a *convex combination* of $\gamma[t_1], \dots, \gamma[t_n]$. We design the function Inc so that the probability pairs representing non-negative integers are *vertices* (i.e., faces of dimension 0) of a certain convex polytope. Furthermore, we identify a certain closure property preventing the use of t_i such that $\gamma[t_i]$ is *outside* the polytope. Thus, whenever $\gamma[t]$ encodes a non-negative integer and $\gamma[t]$ is a positive convex combination of $\gamma[t_1], \dots, \gamma[t_n]$, we can conclude that every $\gamma[t_i]$ is inside the polytope, and consequently every $\gamma[t_i]$ is *equal* to $\gamma[t]$ because $\gamma[t]$ is a vertex of the polytope. Note that this would not be achievable if $\gamma[t]$

was a one-dimensional vector, i.e., if the counter values were encoded by the probability of satisfying a single path formula.

The details are given in Section 4. The design of *Inc* is also influenced by the need of “implementing” *Inc* and its inverse *Dec* in PCTL. The implementation requires additional non-trivial tricks described in Section 5.

3.3 Constructing the formula $\varphi_{\mathcal{M}}$

The formula $\varphi_{\mathcal{M}}$ expressing the existence of a recurrent computation of \mathcal{M} looks as follows:

$$\varphi_{\mathcal{M}} \equiv Struct \wedge Init \wedge \mathbf{G}_{=1}(\bigwedge_{\ell=1}^m (at_{\ell} \Rightarrow Sim_{\ell})) \wedge Rec$$

where

$$\begin{aligned} Rec &\equiv \mathbf{G}_{=1} \left(\bigwedge_{\ell=1}^m (at_{\ell} \Rightarrow \mathbf{F}_{>0}(\bigvee_{\ell' \in \tau} at_{\ell'})) \right) \\ Sim_{\ell} &\equiv (Zero \Rightarrow \bigvee_{\ell' \in Z^{\ell}} Step_{\ell, \ell'}) \wedge (\neg Zero \Rightarrow \bigvee_{\ell' \in P^{\ell}} Step_{\ell, \ell'}) \end{aligned}$$

The subformula *Struct* enforces certain “structural properties” of the model. *Init* says that the state satisfying $\varphi_{\mathcal{M}}$ represents the initial configuration $(1, 0)$. The subformula *Sim_ℓ* says that if the counter *C* currently holds zero/positive value, then *Step_{ℓ, ℓ'}* holds for some index *ℓ'* of *Z^ℓ* or *P^ℓ*, respectively. *Rec* enforces the existence of a run representing a τ -recurrent computation of \mathcal{M} . See Section 5 for more details.

4 Representing Non-Negative Integers by Points in $(0, 1)^2$

In this section, we show how to represent a non-negative counter value by a pair of quantities, and we design functions modeling the increment and decrement operations on the counter. Our starting point are the results invented for *finite* Markov chains in [6]. Our functions *Inc* and *Dec* are modifications of the functions σ and τ of [6], and the set of points determined by *Inc* has a similar structure as the set of points determined by σ (see Fig. 1 left). The crucial novel ingredient is Theorem 5, which is applicable to *arbitrary* Markov chains, including infinite-state ones.

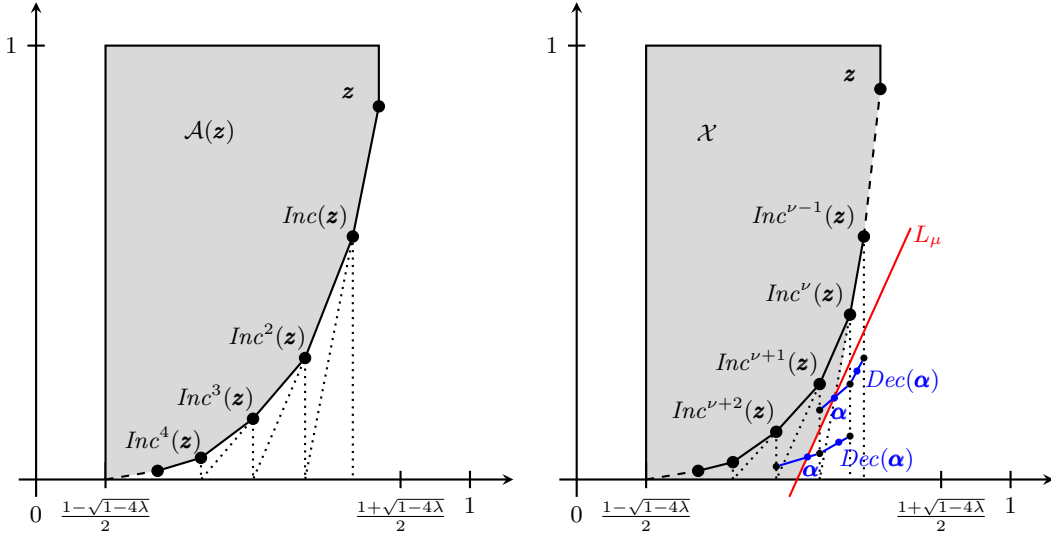
As we already indicated in Section 3, we aim to define suitable $\mathbf{z} \in (0, 1)^2$ and *Inc* : $(0, 1)^2 \rightarrow (0, 1)^2$ such that every $n \in \mathbb{N}$ is encoded by $Inc^n(\mathbf{z})$, where Inc^n is the *n*-fold composition of *Inc*. The points $\mathbf{z}, Inc(\mathbf{z}), Inc^2(\mathbf{z}), \dots$ should be vertices of a certain convex polytope, and both *Inc* and its inverse *Dec* should be “expressible in PCTL”. Roughly speaking, the second condition means that if we have a state *s* representing a configuration (ℓ, c_1, c_2) , then there exists a PCTL formula enforcing the existence of a successor of *s* encoding the values obtained from c_1, c_2 by performing the instructions $update_1^{\ell}$ and $update_2^{\ell}$. After many unsuccessful trials, we have identified a function satisfying these requirements. We put

$$Inc(\mathbf{v}) = \left(\frac{\lambda}{1 - \mathbf{v}_1}, \frac{\mathbf{v}_2 \cdot \lambda}{1 - \mathbf{v}_1} \right)$$

where $\lambda \in (0, \frac{1}{4})$ is a fixed constant. In Section 5, we set λ to a specific value. All observations presented in this section are valid for an *arbitrary* $\lambda \in (0, \frac{1}{4})$.

At first glance, *Inc* is *not* a good choice because it is not even a function $(0, 1)^2 \rightarrow (0, 1)^2$. However, there exists a *subinterval* $I_{\lambda} \subseteq (0, 1)$ such that $Inc : I_{\lambda} \times (0, 1) \rightarrow I_{\lambda} \times (0, 1)$. Furthermore, let

$$Dec(\mathbf{v}) = \left(\frac{\mathbf{v}_1 - \lambda}{\mathbf{v}_1}, \frac{\mathbf{v}_2}{\mathbf{v}_1} \right).$$



■ **Figure 1** The structure of $z, Inc(z), Inc^2(z), \dots$ and $\mathcal{A}(z)$ (left); the construction proving $Out = \emptyset$ (right).

The properties of Inc and Dec are summarized in Lemma 3.

The *slope* of a line $Line$ in \mathbb{R}^2 , denoted by $slope(Line)$, is defined in the standard way. For all $\mathbf{v}, \mathbf{u} \in \mathbb{R}^2$ where $\mathbf{v}_1 \neq \mathbf{u}_1$, we use $slope(\mathbf{v}, \mathbf{u})$ to denote the slope of the line containing \mathbf{v}, \mathbf{u} .

► **Lemma 3.** Let $\lambda \in (0, \frac{1}{4})$ and $I_{\lambda} = \left(\frac{1-\sqrt{1-4\lambda}}{2}, \frac{1+\sqrt{1-4\lambda}}{2} \right)$. For every $\mathbf{v} \in I_{\lambda} \times [0, 1]$ we have the following:

- (a) $Inc(\mathbf{v}) \in I_{\lambda} \times [0, 1]$;
- (b) $Dec(Inc(\mathbf{v})) = \mathbf{v}$;
- (c) $Inc(\mathbf{v})_1 < \mathbf{v}_1$ and $Inc(\mathbf{v})_2 \leq \mathbf{v}_2$; if $\mathbf{v}_2 > 0$, then $Inc(\mathbf{v})_2 < \mathbf{v}_2$;
- (d) let $\mathbf{u} = (Inc^2(\mathbf{v})_1, 0)$; then $slope(\mathbf{u}, Inc(\mathbf{v})) = slope(Inc(\mathbf{v}), \mathbf{v})$;
- (e) let $\mathbf{u} = (Inc(\mathbf{v})_1, y)$ where $0 \leq y < Inc(\mathbf{v})_2$. Then $slope(\mathbf{u}, Dec(\mathbf{u})) < slope(Inc(\mathbf{v}), \mathbf{v})$;
- (f) if \mathbf{u} belongs to the line segment between $Inc^2(\mathbf{v})$ and $Inc(\mathbf{v})$, then $Dec(\mathbf{u})$ belongs to the line segment between $Inc(\mathbf{v})$ and \mathbf{v} ;
- (g) $\lim_{n \rightarrow \infty} Inc^n(\mathbf{v})_1 = \frac{1-\sqrt{1-4\lambda}}{2}$.

Observe that by choosing a sufficiently small $\lambda > 0$, the extreme points of I_{λ} can be pushed to values arbitrarily close to 0 and 1. Let us fix some $\mathbf{z} \in I_{\lambda} \times (0, 1)$. The structure of $\mathbf{z}, Inc(\mathbf{z}), Inc^2(\mathbf{z}), \dots$ is shown in Fig. 1 (left), where the dotted lines illustrate the property of Lemma 3 (d). Furthermore, Fig. 1 (left) also shows the convex polytope $\mathcal{A}(\mathbf{z})$ whose boundaries are the vertical lines going through the points $(\frac{1-\sqrt{1-4\lambda}}{2}, 0)$ and \mathbf{z} , the horizontal line going through the point $(0, 1)$, and all line segments between the consecutive points $Inc^i(\mathbf{z})$ and $Inc^{i+1}(\mathbf{z})$. Note that $\mathcal{A}(\mathbf{z})$ is indeed convex by Lemma 3. Furthermore, each $Inc^i(\mathbf{z})$ is a *vertex* (i.e., a *face* of dimension 0) of $\mathcal{A}(\mathbf{z})$. In our proof of Theorem 5, we also need the following lemma:

► **Lemma 4.** Let $\lambda \in (0, \frac{1}{4})$ and $\mathbf{z} \in I_{\lambda} \times (0, 1)$. For all $\mathbf{v} \in I_{\lambda} \times [0, 1] \setminus \mathcal{A}(\mathbf{z})$ and $n \in \mathbb{N}$ such that $Inc^{n+1}(\mathbf{z})_1 \leq \mathbf{v}_1 \leq Inc^n(\mathbf{z})_1$, there exists \mathbf{u} such that $\mathbf{u}_1 = Inc^{n+1}(\mathbf{z})_1$, $0 \leq \mathbf{u}_2 < Inc^{n+1}(\mathbf{z})_2$, and \mathbf{v} belongs to the line segment between \mathbf{u} and $Dec(\mathbf{u})$.

The next theorem is a crucial tool enabling the construction of the formula $Step_{\ell, \ell'}$ (see Section 3.2).

► **Theorem 5.** *Let $\lambda \in (0, \frac{1}{4})$ and $\mathbf{z} \in I_\lambda \times (0, 1)$. Furthermore, let T be a finite or countably infinite subset of $(\frac{1-\sqrt{1-4\lambda}}{2}, \mathbf{z}_1) \times (0, 1)$ such that for every $\mathbf{v} \in T \setminus \mathcal{A}(\mathbf{z})$, the pair $\text{Dec}(\mathbf{v})$ is a convex combination of the elements of T . Then $T \subseteq \mathcal{A}(\mathbf{z})$.*

Proof. TOPROVE 0 ◀

Observe that if T is a set satisfying the assumptions of Theorem 5, then for every $n \in \mathbb{N}$ we have that if $\text{Inc}^n(\mathbf{z}) = \sum_{\mathbf{v} \in T} \kappa(\mathbf{v}) \cdot \mathbf{v}$ for some probability distribution κ over T , then $\mathbf{v} = \text{Inc}^n(\mathbf{z})$ for all $\mathbf{v} \in T$ such that $\kappa(\mathbf{v}) > 0$ (because $T \subseteq \mathcal{A}(\mathbf{z})$ and $\text{Inc}^n(\mathbf{z})$ is a vertex of $\mathcal{A}(\mathbf{z})$). When constructing the formula $\varphi_{\mathcal{M}}$ (see Section 5), we ensure that for all $k \in \{1, 2\}$, the set of all $\gamma^k[t]$, where t ranges over a set of “relevant” states, satisfies the conditions of Theorem 5. Thus, we overcome the obstacle mentioned in Section 3.1.

5 Constructing the Formula $\varphi_{\mathcal{M}}$

In this section, we show how to construct the formula $\varphi_{\mathcal{M}}$ simulating a given non-deterministic one-counter machine \mathcal{M} with m instructions. The extension to two-counter machines is relatively simple and it is presented in Section 6.

When explaining the meaning of the constructed subformulae of $\varphi_{\mathcal{M}}$, we often refer to some unspecified model of $\varphi_{\mathcal{M}}$. Without restrictions, we assume that all states of the considered model are reachable from a state satisfying $\varphi_{\mathcal{M}}$. The structure of $\varphi_{\mathcal{M}}$ has already been presented in Section 3. Recall that

$$\varphi_{\mathcal{M}} \equiv \text{Struct} \wedge \text{Init} \wedge \mathbf{G}_{=1}(\bigwedge_{\ell=1}^m (\text{at}_\ell \Rightarrow \text{Sim}_\ell)) \wedge \text{Rec}$$

where

$$\begin{aligned} \text{Rec} &\equiv \mathbf{G}_{=1} \left(\bigwedge_{\ell=1}^m (\text{at}_\ell \Rightarrow \mathbf{F}_{>0}(\bigvee_{\ell' \in \tau} \text{at}_{\ell'})) \right) \\ \text{Sim}_\ell &\equiv (\text{Zero} \Rightarrow \bigvee_{\ell' \in Z^\ell} \text{Step}_{\ell, \ell'}) \wedge (\neg \text{Zero} \Rightarrow \bigvee_{\ell' \in P^\ell} \text{Step}_{\ell, \ell'}) \end{aligned}$$

We show how to implement the subformulae of $\varphi_{\mathcal{M}}$, where $\text{Step}_{\ell, \ell'}$ is the main challenge.

Recall that the constant λ of Section 4 can be arbitrarily small. For our purposes, we need to choose λ , \mathbf{z}_1 , \mathbf{z}_2 , and δ (which exists for technical reasons) so that the extreme points of I_λ are rational, $\mathbf{z}_2 < \mathbf{z}_1 < \delta$, and $2\lambda + 2\delta + 2\mathbf{z}_1 + 2\mathbf{z}_2 < 1$. For example, we can put $\lambda = \frac{14}{255}$, $I_\lambda = (\frac{1}{15}, \frac{14}{15})$, $\mathbf{z} = (\frac{1}{12}, \frac{1}{15})$, and $\delta = \frac{1}{11}$. Furthermore, let

$$\begin{aligned} \mathcal{B} &= \{K\} \cup \{A_i, B_i, C_i, D_i, E_i, R_i \mid 0 \leq i \leq 2\} \\ \mathcal{A} &= \mathcal{B} \cup \{a_{i, \ell}, \bar{a}_{i, \ell}, b_{i, \ell}, c_{i, \ell}, d_{i, \ell}, r_{i, \ell} \mid 0 \leq i \leq 2, 1 \leq \ell \leq m\} \end{aligned}$$

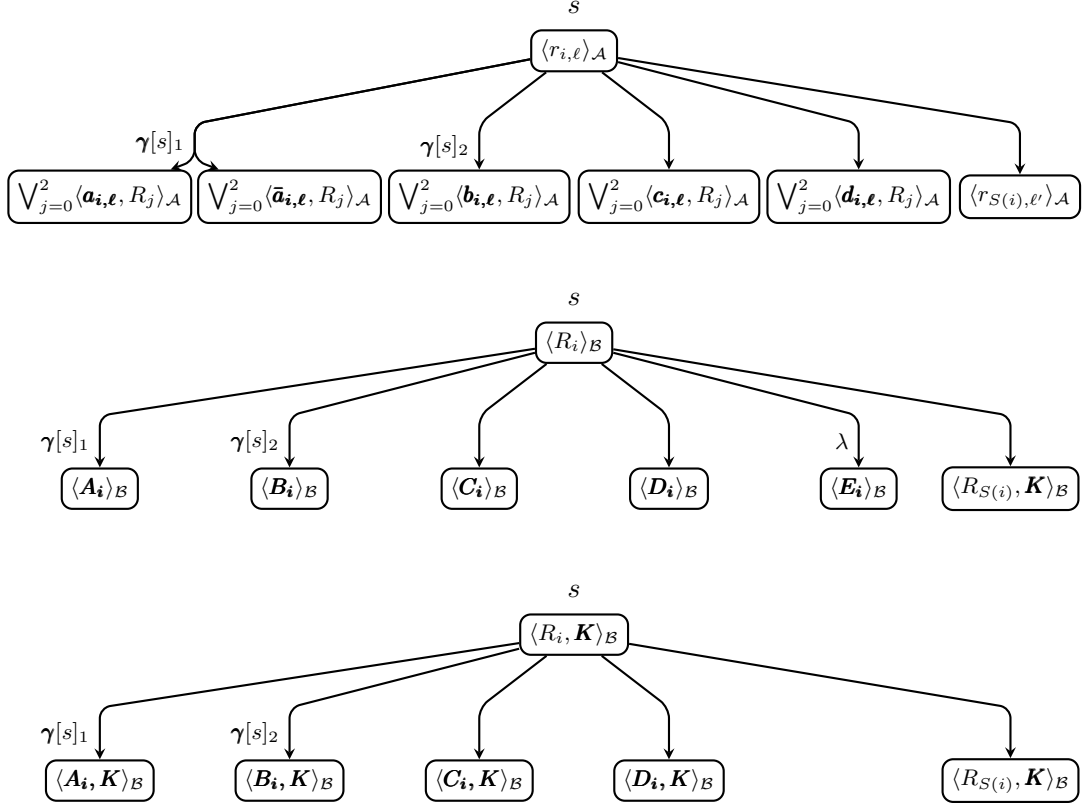
be sets of atomic propositions. For every $i \in \{0, 1, 2\}$, we use $S(i)$ to denote $i+1 \bmod 3$, and for all $O \subseteq \mathcal{A}$ and $L \subseteq O$, let $\langle L \rangle_O \equiv \bigwedge_{p \in L} p \wedge \bigwedge_{p \in O \setminus L} \neg p$.

5.1 Defining the basic structure of a model

Intuitively, states satisfying $r_{i, \ell}$ are the states where $\varphi_{\mathcal{M}}$ simulates the instruction Ins_ℓ of \mathcal{M} . The role of the index i in $r_{i, \ell}$ is auxiliary. For every $1 \leq \ell \leq m$, we put $\text{at}_\ell \equiv \langle r_{0, \ell} \rangle_{\mathcal{A}} \vee \langle r_{1, \ell} \rangle_{\mathcal{A}} \vee \langle r_{2, \ell} \rangle_{\mathcal{A}}$. As we shall see, a state satisfying $\varphi_{\mathcal{M}}$ satisfies $\langle r_{0, 1} \rangle_{\mathcal{A}}$.

The formula Struct defines the basic structure of a model of $\varphi_{\mathcal{M}}$. We put

$$\text{Struct} \equiv \text{Succ} \wedge \text{Mark} \wedge \text{Lambda}$$



■ **Figure 2** Illustrating the meaning of *Struct*.

where

$$\begin{aligned}
 Succ &\equiv \bigwedge_{\ell=1}^m \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(\langle r_{i,\ell} \rangle_{\mathcal{A}} \Rightarrow \bigvee_{\ell'=1}^m \langle r_{i,\ell} \rangle_{\mathcal{A}} \mathbf{U}_{=1} rsuc_{i,\ell,\ell'} \right) \\
 &\wedge \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left((\langle R_i \rangle_{\mathcal{B}} \Rightarrow \langle R_i \rangle_{\mathcal{B}} \mathbf{U}_{=1} Rsuc_i) \wedge (\langle R_i, \mathbf{K} \rangle_{\mathcal{B}} \Rightarrow \langle R_i, \mathbf{K} \rangle_{\mathcal{B}} \mathbf{U}_{=1} RKsuc_i) \right)
 \end{aligned}$$

where

$$\begin{aligned}
 rsuc_{i,\ell,\ell'} &\equiv \langle r_{S(i),\ell'} \rangle_{\mathcal{A}} \vee \bigvee_{j=0}^2 (\langle \mathbf{a}_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle \bar{\mathbf{a}}_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle \mathbf{b}_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle \mathbf{c}_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle \mathbf{d}_{i,\ell}, R_j \rangle_{\mathcal{A}}) \\
 Rsuc_i &\equiv \langle \mathbf{A}_i \rangle_{\mathcal{B}} \vee \langle \mathbf{B}_i \rangle_{\mathcal{B}} \vee \langle \mathbf{C}_i \rangle_{\mathcal{B}} \vee \langle \mathbf{D}_i \rangle_{\mathcal{B}} \vee \langle \mathbf{E}_i \rangle_{\mathcal{B}} \vee \langle R_{S(i)}, \mathbf{K} \rangle_{\mathcal{B}} \\
 RKsuc_i &\equiv \langle \mathbf{A}_i, \mathbf{K} \rangle_{\mathcal{B}} \vee \langle \mathbf{B}_i, \mathbf{K} \rangle_{\mathcal{B}} \vee \langle \mathbf{C}_i, \mathbf{K} \rangle_{\mathcal{B}} \vee \langle \mathbf{D}_i, \mathbf{K} \rangle_{\mathcal{B}} \vee \langle R_{S(i)}, \mathbf{K} \rangle_{\mathcal{B}}
 \end{aligned}$$

Let $O \subseteq \mathcal{A}$ and $L \subseteq O$. We say that L is an O marker if whenever $s \models \langle L \rangle_O$, then $s' \models \langle L \rangle_O$ for every s' reachable from s . Note that this property can be encoded by the PCTL formula $\langle L \rangle_O \Rightarrow \mathbf{G}_{=1} \langle L \rangle_O$. The formula *Mark* says that

- the sets $\{a_{i,\ell}\}$, $\{\bar{a}_{i,\ell}\}$, $\{b_{i,\ell}\}$, $\{c_{i,\ell}\}$, and $\{d_{i,\ell}\}$ are $\mathcal{A} \setminus \mathcal{B}$ markers,
- the sets $\{\mathbf{A}_i\}$, $\{\mathbf{A}_i, \mathbf{K}\}$, $\{\mathbf{B}_i\}$, $\{\mathbf{B}_i, \mathbf{K}\}$, $\{\mathbf{C}_i\}$, $\{\mathbf{C}_i, \mathbf{K}\}$, $\{\mathbf{D}_i\}$, $\{\mathbf{D}_i, \mathbf{K}\}$, and $\{\mathbf{E}_i\}$ are \mathcal{B} markers.

Finally, $Lambda \equiv \bigwedge_{i=0}^2 (\langle R_i \rangle_{\mathcal{B}} \Rightarrow \mathbf{G}_{=\lambda} (R_i \vee E_i))$. The purpose of *Lambda* is clarified when constructing the formula $Step_{\ell,\ell'}$.

The meaning of *Struct* is illustrated in Fig. 2. The first subformula of *Succ* guarantees that if $s \models \langle r_{i,\ell} \rangle_{\mathcal{A}}$, then for almost all $\pi \in \text{Run}(s)$ there is $n \in \mathbb{N}$ such that

- $\pi(n') \models \langle r_{i,\ell} \rangle_{\mathcal{A}}$ for every $0 \leq n' < n$;
- $\pi(n)$ satisfies either $\bigvee_{j=0}^2 \langle a_{i,\ell}, R_j \rangle_{\mathcal{A}}$, or $\bigvee_{j=0}^2 \langle \bar{a}_{i,\ell}, R_j \rangle_{\mathcal{A}}$, or $\bigvee_{j=0}^2 \langle b_{i,\ell}, R_j \rangle_{\mathcal{A}}$, or $\bigvee_{j=0}^2 \langle c_{i,\ell}, R_j \rangle_{\mathcal{A}}$, or $\bigvee_{j=0}^2 \langle d_{i,\ell}, R_j \rangle_{\mathcal{A}}$, or $\langle r_{S(i),\ell'} \rangle_{\mathcal{A}}$.

This is shown in Fig. 2, together with the properties of states satisfying $\langle R_i \rangle_{\mathcal{B}}$ and $\langle R_i, K \rangle_{\mathcal{B}}$ enforced by the other subformulae of *Succ*. The markers are typeset in boldface, and the probabilities constituting $\gamma[s]_1$ and $\gamma[s]_2$ are also shown. The meaning of *Lambda* is apparent in Fig. 2 (middle).

The three graphs of Fig. 2 describe the structure of every model of $\varphi_{\mathcal{M}}$. As we shall see, $s \models \varphi_{\mathcal{M}}$ implies $s \models \langle r_{0,1} \rangle_{\mathcal{A}}$. Almost every run initiated in s eventually visits a state s' satisfying one of the six formulae shown in Fig. 2 (top). For example, suppose that $s' \models \langle b_{i,\ell}, R_j \rangle_{\mathcal{B}}$. Since $b_{i,\ell}$ is a marker, this proposition is valid in all successors of s' . Furthermore, $s' \models \langle R_j \rangle_{\mathcal{B}}$, and the structure of states reachable from s' is described in Fig. 2 (middle), i.e., almost every run initiated in s' eventually visits a state s'' satisfying one of the six formulae shown in Fig. 2 (middle). If s'' satisfies one of the first five formulae, the states reachable from s'' are no longer interesting (all of them satisfy the associated \mathcal{B} marker). If $s'' \models \langle R_{S(j)}, K \rangle_{\mathcal{B}}$, the structure of states reachable from s'' is shown in Fig. 2 (bottom).

The most interesting runs initiated in a state s such that $s \models \varphi_{\mathcal{M}}$ are the runs visiting infinitely many states satisfying a formula of the form $\langle r_{i,\ell} \rangle_{\mathcal{A}}$ (in Fig. 2 (top), keep following the rightmost branch). These runs are responsible for modelling the computations of \mathcal{M} .

5.2 Encoding the counter value

Now we define the path formulae Φ_s and Ψ_s encoding the current value of the counter C (see Section 3.1). Let $\mathcal{R} = \{r_{i,\ell}, R_i \mid 0 \leq i \leq 2, 1 \leq \ell \leq m\}$. A state is *relevant* if it satisfies some proposition of \mathcal{R} (note that according to *Struct*, every relevant state satisfies *exactly one* proposition of \mathcal{R}). If $s \models r_{i,\ell}$, we put $\Phi_s \equiv \mathbf{G}(r_{i,\ell} \vee a_{i,\ell} \vee \bar{a}_{i,\ell})$ and $\Psi_s \equiv \mathbf{G}(r_{i,\ell} \vee b_{i,\ell})$. Similarly, if $s \models R_i$, we put $\Phi_s \equiv \mathbf{G}(R_i \vee A_i)$ and $\Psi_s \equiv \mathbf{G}(R_i \vee B_i)$. Furthermore, $\gamma[s] \in [0, 1]^2$ denotes the vector such that $\gamma[s]_1$ and $\gamma[s]_2$ are the probabilities of satisfying Φ_s and Ψ_s in s , respectively. We define T as the set of all $\gamma[s]$ where s is a relevant state. We need to ensure that T satisfies the conditions of Theorem 5. Let

$$\begin{aligned} \text{Zero} &\equiv \bigwedge_{i=0}^2 \bigwedge_{\ell=1}^m \left(r_{i,\ell} \Rightarrow (\mathbf{G}_{=\mathbf{z}_1}(r_{i,\ell} \vee a_{i,\ell} \vee \bar{a}_{i,\ell}) \wedge \mathbf{G}_{=\mathbf{z}_2}(r_{i,\ell} \vee b_{i,\ell})) \right) \\ &\quad \wedge \bigwedge_{i=0}^2 \left(R_i \Rightarrow (\mathbf{G}_{=\mathbf{z}_1}(R_i \vee A_i) \wedge \mathbf{G}_{=\mathbf{z}_2}(R_i \vee B_i)) \right) \\ \text{Eligible} &\equiv \bigwedge_{i=0}^2 \bigwedge_{\ell=1}^m \mathbf{G}_{=1} \left(r_{i,\ell} \Rightarrow (\mathbf{G}_{\leq \mathbf{z}_1}(r_{i,\ell} \vee a_{i,\ell} \vee \bar{a}_{i,\ell}) \wedge \mathbf{G}_{\geq \beta}(r_{i,\ell} \vee a_{i,\ell} \vee \bar{a}_{i,\ell})) \right) \\ &\quad \wedge \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(R_i \Rightarrow (\mathbf{G}_{\leq \mathbf{z}_1}(R_i \vee A_i) \wedge \mathbf{G}_{\geq \beta}(R_i \vee A_i)) \right) \end{aligned}$$

where β is the lower extreme point of I_λ (recall that λ is chosen so that β is rational). Hence, for every relevant state s , we have that $s \models \text{Zero}$ iff $\gamma[s] = \mathbf{z}$. The formula *Eligible* says that $\gamma[s]_1 \in ((1 - \sqrt{1 - 4\lambda})/2, \mathbf{z}_1)$ for every relevant s , i.e., $T \subseteq ((1 - \sqrt{1 - 4\lambda})/2, \mathbf{z}_1) \times (0, 1)$. The closure property of Theorem 5 is enforced by the formulae constructed in the next paragraph.

5.3 The subformula *Init*

We put

$$Init \equiv \langle r_{0,1} \rangle_{\mathcal{A}} \wedge Zero \wedge Eligible \wedge \mathbf{G}_{=1} Decrement$$

The subformula $\langle r_{0,1} \rangle_{\mathcal{A}} \wedge Zero$ says that a state satisfying $\varphi_{\mathcal{M}}$ represents the initial configuration $(1, 0)$ of \mathcal{M} . The subformulae *Eligible* and $\mathbf{G}_{=1} Decrement$, together with *Step_{ℓ,ℓ'}* constructed below, ensure that the set T introduced in the previous section satisfies the assumptions of Theorem 5 (as we shall see in Section 5.4, *Decrement* is similar to the formula implementing the *dec* operation on a counter).

We put

$$Decrement \equiv \bigwedge_{i=0}^2 (R_i \wedge \neg Zero) \Rightarrow \left(Copy_i \wedge Succ_i \right)$$

where

$$\begin{aligned} Copy_i &\equiv \mathbf{G}_{=\delta} (R_i \vee A_i \vee C_i) \wedge \mathbf{F}_{=\delta} (R_{S(i)} \vee C_i) \\ Succ_i &\equiv \mathbf{F}_{=\lambda} (B_{S(i)} \vee C_{S(i)} \vee D_{S(i)} \vee R_{S^2(i)}) \wedge \mathbf{F}_{=\lambda} (B_i \vee C_{S(i)} \vee D_{S(i)} \vee R_{S^2(i)}) \end{aligned}$$

To explain the meaning of *Decrement*, suppose that $s \models R_i \wedge Decrement$, where s is a state reachable from a state satisfying $\varphi_{\mathcal{M}}$ (hence, $s \models Struct$). Let U be the set of all states u such that $u \models R_{S(i)} \wedge K$ and $p_u > 0$, where p_u is the probability of all runs w initiated in s such that w visits u and all states preceding this visit satisfy R_i (see Fig. 3). The formula *Decrement* ensures that if $s \models \neg Zero$, then $Dec(\gamma[s]) = \sum_{u \in U} (p_u/P) \cdot \gamma[u]$, where $P = \sum_{u \in U} p_u$. Hence, the condition of Theorem 5 is satisfied for $\gamma[s]$.

More specifically, the subformula *Copy_i* “copies” $\gamma[s]_1$ into P , i.e., ensures that $P = \gamma[s]_1$. We claim that

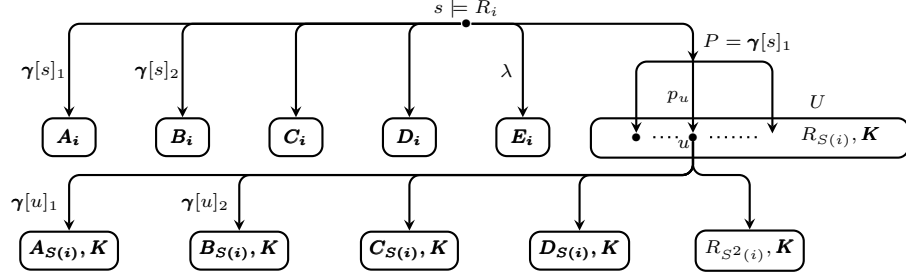
- P is the probability of satisfying the path formula $\mathbf{F} R_{S(i)}$ in s ;
 - $\gamma[s]_1$ is the probability of satisfying the path formula $\mathbf{G}(R_i \vee A_i)$ in s .
- Both claims follow directly from the formula *Struct*. However, the equality $P = \gamma[s]_1$ is not directly expressible in PCTL. Instead, the formula *Copy_i* requires that $\gamma[s]_1 + \kappa = \delta$ and $P + \kappa = \delta$, where κ is the probability of satisfying the formula $\mathbf{G}(R_i \vee C_i)$ in s .
- $P + \kappa$ is the probability of satisfying the formula $\mathbf{F}(R_{S(i)} \vee C_i)$ in s .
 - $\gamma[s]_1 + \kappa$ is the probability of satisfying the formula $\mathbf{G}(R_i \vee A_i \vee C_i)$ in s .

The first conjunct of *Succ_i* says that the probability of satisfying $\mathbf{F}(B_{S(i)} \vee C_{S(i)} \vee D_{S(i)} \vee R_{S^2(i)})$ in s is equal to λ . By inspecting the structure of states reachable from s , we obtain that the above formula is satisfied with the probability $\sum_{u \in U} p_u \cdot (1 - \gamma[u]_1)$. Hence, $\sum_{u \in U} p_u \cdot (1 - \gamma[u]_1) = \lambda$ which implies $(P - \lambda)/P = \sum_{u \in U} (p_u/P) \cdot \gamma[u]_1$. Since $P = \gamma[s]_1$, we have that

$$Dec(\gamma[s])_1 = (\gamma[s]_1 - \lambda) / \gamma[s]_1 = \sum_{u \in U} (p_u/P) \cdot \gamma[u]_1.$$

The second conjunct of *Succ_i* says that the probability of satisfying $\mathbf{F}(B_i \vee C_{S(i)} \vee D_{S(i)} \vee R_{S^2(i)})$ in s is also equal to λ , i.e., it is the *same* as the probability of satisfying $\mathbf{F}(B_{S(i)} \vee C_{S(i)} \vee D_{S(i)} \vee R_{S^2(i)})$. From this we obtain that the following two probabilities are also equal:

- The probability of all runs π initiated in s visiting a state satisfying B_i so that all states in π preceding this visit do not satisfy $R_{S^2(i)}$.



■ **Figure 3** Illustrating the meaning of *Decrement*.

- The probability of all runs π initiated in s visiting a state satisfying $B_{S(i)}$ so that all states in π preceding this visit do not satisfy $R_{S^2(i)}$.

The first probability is equal to $\gamma[s]_2$, and the second probability is equal to $\sum_{u \in U} p_u \cdot \gamma[u]_2$. Hence, $\gamma[s]_2 = \sum_{u \in U} p_u \cdot \gamma[u]_2$. Since $P = \gamma[s]_1$ (see above), we obtain

$$Dec(\gamma[s])_2 = \frac{\gamma[s]_2}{\gamma[s]_1} = \frac{\sum_{u \in U} p_u \cdot \gamma[u]_2}{P} = \sum_{u \in U} (p_u/P) \cdot \gamma[u]_2.$$

To sum up, $Dec(\gamma[s]) = \sum_{u \in U} (p_u/P) \cdot \gamma[u]$ is a convex combination of $\gamma[u]$ where $u \in U$.

5.4 The subformula Sim_ℓ

Recall that

$$Sim_\ell \equiv (Zero \Rightarrow \bigvee_{\ell' \in Z^\ell} Step_{\ell, \ell'}) \wedge (\neg Zero \Rightarrow \bigvee_{\ell' \in P^\ell} Step_{\ell, \ell'})$$

where

$$Step_{\ell, \ell'} \equiv \bigwedge_{i=0}^2 \langle r_{i, \ell} \rangle_{\mathcal{A}} \Rightarrow (\langle r_{i, \ell} \rangle_{\mathcal{A}} \mathbf{U}_{=1} rsuc_{i, \ell, \ell'} \wedge Update_{i, \ell, \ell'})$$

The purpose of the $\langle r_{i, \ell} \rangle_{\mathcal{A}} \mathbf{U}_{=1} rsuc_{i, \ell, \ell'}$ subformula of $Step_{\ell, \ell'}$ is to ensure that the ℓ' index is “propagated” to the states reachable from a state satisfying $Step_{\ell, \ell'}$ (recall that *Struct* ensures the propagation of *some* counter index which may be different from ℓ').

To explain the meaning of $Step_{\ell, \ell'}$ and the formula $Update_{i, \ell}$ constructed below, let us assume that $s \models Step_{\ell, \ell'}$ where s is a state reachable from a state satisfying $\varphi_{\mathcal{M}}$ (in particular, $s \models Struct$). We distinguish two subcases.

Ins_ℓ updates the counter by *dec*. In this case, $Update_{i, \ell, \ell'}$ is similar to *Decrement*. We define the set U of all states u such that $u \models r_{S(i), \ell'}$ and $p_u > 0$, where p_u is the probability of all runs π initiated in s such that π visits u and all states preceding this visit satisfy $r_{i, \ell}$.

The formula $Update_{i, \ell, \ell'}$ ensures that if $s \models \neg Zero$, then $Dec(\gamma[s])$ is a positive convex combination of $\gamma[u]$ where $u \in U$. In addition, it ensures that if $s \models Zero$, then \mathbf{z} is a positive convex combination of $\gamma[u]$ where $u \in U$. We put $Update_{i, \ell, \ell'} \equiv UDec_{i, \ell, \ell'}$ where

$$UDec_{i, \ell, \ell'} \equiv Zero \Rightarrow (\mathbf{G}_{=\mathbf{z}_1} (r_{i, \ell} \vee r_{S(i), \ell'} \vee a_{S(i), \ell'}) \wedge \mathbf{G}_{=\mathbf{z}_2} (r_{i, \ell} \vee r_{S(i), \ell'} \vee b_{S(i), \ell'})) \quad (2)$$

$$\wedge \neg Zero \Rightarrow (UCopy_{i, \ell, \ell'} \wedge USucc_{i, \ell, \ell'}) \quad (3)$$

where

$$UCopy_{i,\ell,\ell'} \equiv \mathbf{G}_{=\delta}(r_{i,\ell} \vee a_{i,\ell} \vee \bar{a}_{i,\ell} \vee c_{i,\ell}) \wedge \mathbf{F}_{=\delta}(r_{S(i),\ell'} \vee c_{i,\ell})$$

$$\begin{aligned} USucc_{i,\ell,\ell'} &\equiv \mathbf{F}_{=\lambda}(b_{S(i),\ell'} \vee c_{S(i),\ell'} \vee d_{S(i),\ell'} \vee \bigvee_{\ell''=1}^m r_{S^2(i),\ell''}) \\ &\wedge \mathbf{F}_{=\lambda}(b_{i,\ell} \vee c_{S(i),\ell'} \vee d_{S(i),\ell'} \vee \bigvee_{\ell''=1}^m r_{S^2(i),\ell''}) \end{aligned}$$

By using similar arguments as in Section 5.3, it is easy to verify that

- if $s \models UDec_{i,\ell,\ell'} \wedge \neg Zero$, then $Dec(\gamma[s]) = \sum_{u \in U} (p_u/P) \cdot \gamma[u]$;
- if $s \models UDec_{i,\ell,\ell'} \wedge Zero$, then $\mathbf{z} = \sum_{u \in U} (p_u/P) \cdot \gamma[u]$.

***Inc_ℓ* updates the counter by *inc*.**

Let V be the set consisting of all v such that $v \models a_{i,\ell} \vee \bar{a}_{i,\ell}$ and $q_v > 0$, where q_v is the probability of all runs π initiated in s such that π visits v and all states preceding this visit satisfy $r_{i,\ell}$. Furthermore, let W be the set consisting of all w such that $w \models b_{i,\ell} \vee c_{i,\ell} \vee d_{i,\ell} \vee r_{S(i),\ell'}$ and $q_w > 0$, where q_w is the probability of all runs π initiated in s such that π visits w and all states preceding this visit satisfy $r_{i,\ell}$. The formula $Update_{i,\ell,\ell'}$ ensures the following:

- $Inc(\gamma[s])$ is a positive convex combination of $\gamma[w]$ where $w \in W$.
 - If $s \models \neg Zero$, then $Dec(\gamma[s])$ is a positive convex combination of $\gamma[v]$ where $v \in V$.
- We put $Update_{i,\ell,\ell'} \equiv UInc_{i,\ell,\ell'}$ where $UInc_{i,\ell,\ell'}$ is the following formula:

$$\mathbf{G}_{=\lambda} \left((r_{i,\ell} \vee \bigvee_{j=0}^2 R_j \vee r_{S(i),\ell'} \vee a_{S(i),\ell'} \vee \bar{a}_{S(i),\ell'} \vee \bigvee_{j=0}^2 A_j) \wedge \neg K \wedge \neg a_{i,\ell} \wedge \neg \bar{a}_{i,\ell} \right) \quad (4)$$

$$\wedge \mathbf{G}_{=\varrho} \left((r_{i,\ell} \vee \bigvee_{j=0}^2 R_j \vee r_{S(i),\ell'} \vee \bar{a}_{i,\ell} \vee \bigvee_{j=0}^2 B_j \vee b_{S(i),\ell'}) \wedge \neg a_{i,\ell} \wedge (K \Rightarrow \bar{a}_{i,\ell}) \right) \quad (5)$$

$$\wedge \mathbf{G}_{=\varrho} \left((r_{i,\ell} \vee \bigvee_{j=0}^2 R_j \vee \bar{a}_{i,\ell} \vee \bigvee_{j=0}^2 (E_j \wedge b_{i,\ell})) \wedge (K \Rightarrow \bar{a}_{i,\ell}) \right) \quad (6)$$

$$\wedge \neg Zero \Rightarrow \mathbf{G}_{=\lambda} (r_{i,\ell} \vee a_{i,\ell} \vee \bar{a}_{i,\ell}) \wedge \bigwedge_{j=0}^2 (A_j \Rightarrow K) \quad (7)$$

$$\wedge \neg Zero \Rightarrow \mathbf{G}_{=\delta} \left((r_{i,\ell} \vee c_{i,\ell} \vee \bigvee_{j=0}^2 (R_j \wedge a_{i,\ell}) \vee \bigvee_{j=0}^2 (R_j \wedge \bar{a}_{i,\ell}) \vee \bigvee_{j=0}^2 B_j) \wedge (K \Rightarrow c_{i,\ell}) \right) \quad (8)$$

$$\wedge \neg Zero \Rightarrow \mathbf{G}_{=\delta} (r_{i,\ell} \vee b_{i,\ell} \vee c_{i,\ell}) \quad (9)$$

The meaning of $UInc_{i,\ell,\ell'}$ is illustrated in Fig. 4. Suppose $s \models UInc_{i,\ell,\ell'}$. Every subformula of $UInc_{i,\ell,\ell'}$ specifies the probability of visiting a state of a certain family by a run initiated in s . In Fig. 4, these families are indicated by colored shapes (some states belong to multiple families).

- The subformula (4) says that the probability of reaching a state of the *yellow circle* family is equal to λ , i.e., $\sum_{w \in W} p_w \cdot \gamma[w]_1 = \lambda$. Since $P_W = \sum_{w \in W} p_w = (1 - \gamma[s]_1)$, we obtain that

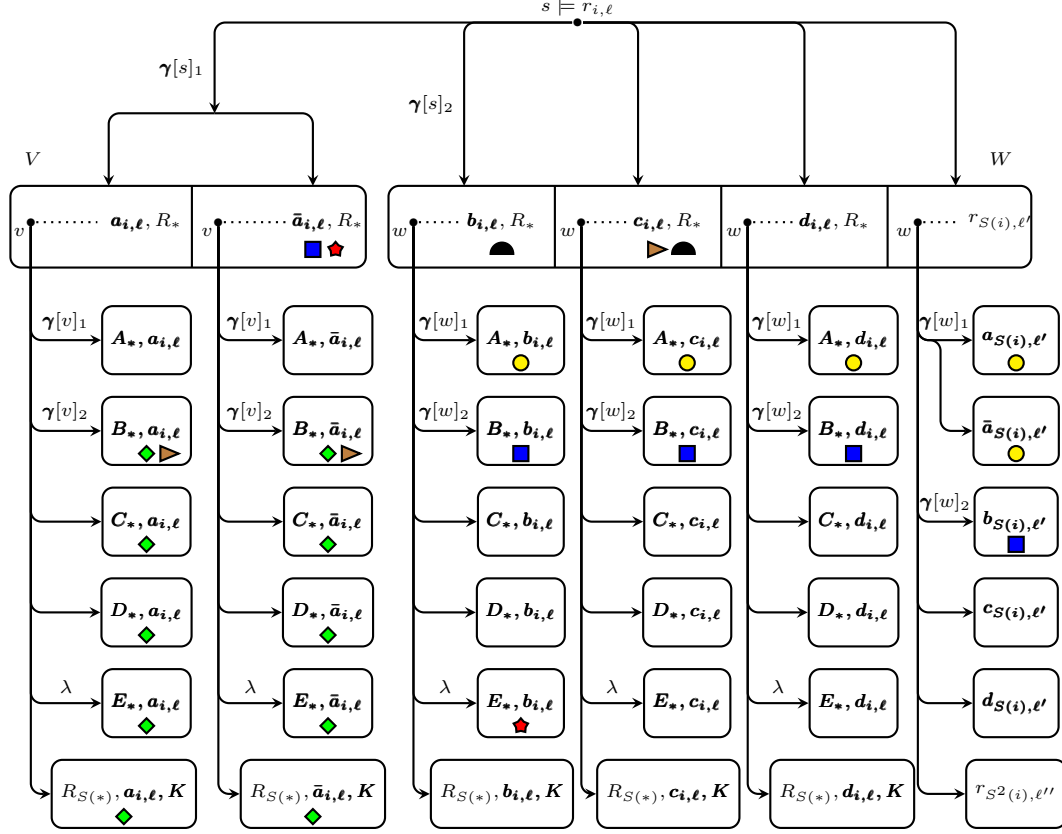
$$Inc(\gamma[s])_1 = \frac{\lambda}{1 - \gamma[s]_1} = \frac{\sum_{w \in W} p_w \cdot \gamma[w]_1}{P_W} = \sum_{w \in W} (p_w/P_W) \cdot \gamma[w]_1$$

- The subformulae (5) and (6) say that the probabilities of reaching a state of the *blue square* and the *red star* families are equal to ϱ . Consequently, $\sum_{w \in W} p_w \cdot \gamma[w]_2 = \gamma[s]_2 \cdot \lambda$ (this also explains the purpose of the subformula *Lambda*, see Section 5.1). This implies

$$Inc(\gamma[s])_2 = \frac{\gamma[s]_2 \cdot \lambda}{1 - \gamma[s]_1} = \frac{\sum_{w \in W} p_w \cdot \gamma[w]_2}{P_W} = \sum_{w \in W} (p_w/P_W) \cdot \gamma[w]_2$$

- If $s \models \neg Zero$, then the subformula (7) says that the probability of reaching a state of the *green diamond* family is equal to λ . Hence, $\sum_{v \in V} p_v \cdot (1 - \gamma[v]_1) = \lambda$. Since $P_V = \sum_{v \in V} p_v = \gamma[s]_1$, we obtain

$$Dec(\gamma[s])_1 = \frac{\gamma[s]_1 - \lambda}{\gamma[s]_1} = \frac{\gamma[s]_1 - \sum_{v \in V} p_v \cdot (1 - \gamma[v]_1)}{P_V} = \sum_{v \in V} (p_v/P_V) \cdot \gamma[v]_1$$



■ **Figure 4** Illustrating the meaning of $Update_{i, \ell, \ell'}$ when Ins_{ℓ} updates the counter by inc . The markers are in boldface, and ‘*’ indicates an index $j \in \{0, 1, 2\}$. The colored shapes indicate the families of states used in the subformulae of $UInc_{i, \ell, \ell'}$. Note that the probability of certain transitions is equal to λ due to the subformula $Lambda$.

- If $s \models \neg Zero$, then the subformulae (8) and (9) say that the probabilities of reaching a state of the *brown triangle* and the *black semicircle* family are equal to δ . This implies $\sum_{v \in V} p_v \cdot \gamma[v]_2 = \gamma[s]_2$. Hence,

$$Dec(\gamma[s])_2 = \frac{\gamma[s]_2}{\gamma[s]_1} = \frac{\sum_{v \in V} p_v \cdot \gamma[v]_2}{P_V} = \sum_{v \in V} (p_v / P_V) \cdot \gamma[v]_2.$$

5.5 Justifying the correctness of $\varphi_{\mathcal{M}}$

We claim that $\varphi_{\mathcal{M}}$ is satisfiable iff \mathcal{M} has a recurrent computation, i.e.,

- if $t \models \varphi_{\mathcal{M}}$, then there is a run π initiated in t representing a recurrent computation of \mathcal{M} (here we use the results of Section 4);
- for every recurrent computation of \mathcal{M} , there is a model of $\varphi_{\mathcal{M}}$ containing a run representing the computation.

We do not give explicit proofs for these claims since they are simplified versions of the proofs of Propositions 6 and 7 formulated in Section 6.

6 Extension to Non-Deterministic Two-Counter Machines

Now we show how to adapt the construction of $\varphi_{\mathcal{M}}$ presented in Section 5 when \mathcal{M} is a non-deterministic *two-counter* machine with m instructions.

Let $\mathcal{M}^1, \mathcal{M}^2$ be non-deterministic one-counter machines obtained from \mathcal{M} by changing every instruction of form

$$\langle C_k=0? Z : P \rangle : \text{update}_1, \text{update}_2$$

into

$$\langle C=0? Z : P \rangle : \text{update}$$

where $\text{update} \equiv \text{update}_1$ for \mathcal{M}^1 and $\text{update} \equiv \text{update}_2$ for \mathcal{M}^2 . Observe that \mathcal{M}^1 and \mathcal{M}^2 do *not* simulate \mathcal{M} in any reasonable sense.

Consider the formulae $\varphi_{\mathcal{M}^1}$ and $\varphi_{\mathcal{M}^2}$ constructed for \mathcal{M}^1 and \mathcal{M}^2 in the way described in the previous sections, where the underlying sets \mathcal{A}^1 and \mathcal{A}^2 of atomic propositions are disjoint. For every subformula *Form* constructed in the previous sections, we use Form^1 and Form^2 to denote the corresponding formulae of $\varphi_{\mathcal{M}^1}$ and $\varphi_{\mathcal{M}^2}$, respectively.

Intuitively, the formula $\varphi_{\mathcal{M}}$ is “basically” the conjunction of $\varphi_{\mathcal{M}^1}$ and $\varphi_{\mathcal{M}^2}$ with some additional synchronization. First, for every $\ell \in \{1, \dots, m\}$, let NewZero_ℓ be the formula defined as follows:

- If the counter tested for zero in Ins_ℓ is C_1 , then $\text{NewZero}_\ell \equiv \text{Zero}^1$.
- If the counter tested for zero in Ins_ℓ is C_2 , then $\text{NewZero}_\ell \equiv \text{Zero}^2$.

Recall that $\varphi_{\mathcal{M}^1}$ and $\varphi_{\mathcal{M}^2}$ contain the subformulae Sim_ℓ^1 and Sim_ℓ^2 corresponding to the subformula

$$\text{Sim}_\ell \equiv (\text{Zero} \Rightarrow \bigvee_{\ell' \in Z^\ell} \text{Step}_{\ell, \ell'}) \wedge (\neg \text{Zero} \Rightarrow \bigvee_{\ell' \in P^\ell} \text{Step}_{\ell, \ell'})$$

defined in Section 5.4. Let $\psi_{\mathcal{M}^1}$ and $\psi_{\mathcal{M}^2}$ be the formulae obtained from $\varphi_{\mathcal{M}^1}$ and $\varphi_{\mathcal{M}^2}$ by replacing the two Sim_ℓ^1 and Sim_ℓ^2 with NewSim_ℓ^1 and NewSim_ℓ^2 , where

$$\begin{aligned} \text{NewSim}_\ell^1 &\equiv ((\text{NewZero}_\ell \wedge \text{at}_\ell^2) \Rightarrow \bigvee_{\ell' \in Z^\ell} \text{Step}_{\ell, \ell'}^1) \wedge ((\neg \text{NewZero}_\ell \wedge \text{at}_\ell^2) \Rightarrow \bigvee_{\ell' \in P^\ell} \text{Step}_{\ell, \ell'}^1) \\ &\quad \wedge \bigwedge_{i=0}^2 ((\langle r_{i, \ell}^1 \rangle_{\mathcal{A}^1} \wedge \neg \text{at}_\ell^2) \Rightarrow (\langle r_{i, \ell}^1 \rangle_{\mathcal{A}^1} \mathbf{u}_{=1} \text{rsuc}_{i, \ell, \ell}^1 \wedge \text{UDec}_{i, \ell, \ell}^1)) \\ \text{NewSim}_\ell^2 &\equiv ((\text{NewZero}_\ell \wedge \text{at}_\ell^1) \Rightarrow \bigvee_{\ell' \in Z^\ell} \text{Step}_{\ell, \ell'}^2) \wedge ((\neg \text{NewZero}_\ell \wedge \text{at}_\ell^1) \Rightarrow \bigvee_{\ell' \in P^\ell} \text{Step}_{\ell, \ell'}^2) \\ &\quad \wedge \bigwedge_{i=0}^2 ((\langle r_{i, \ell}^2 \rangle_{\mathcal{A}^2} \wedge \neg \text{at}_\ell^1) \Rightarrow (\langle r_{i, \ell}^2 \rangle_{\mathcal{A}^2} \mathbf{u}_{=1} \text{rsuc}_{i, \ell, \ell}^2 \wedge \text{UDec}_{i, \ell, \ell}^2)) \end{aligned}$$

Intuitively, replacing Zero with NewZero_ℓ ensures that the counter tested for zero in NewSim_ℓ is indeed the counter tested for zero in Ins_ℓ . The reason for adding the third conjunct in NewSim_ℓ^1 and NewSim_ℓ^2 is more subtle. Roughly speaking, we cannot prevent the situation when a state satisfies at_ℓ^1 but not at_ℓ^2 (or vice versa). In this case, it does not make sense to continue the simulation of \mathcal{M} . However, we still need to ensure that the assumptions of Theorem 5 are satisfied. Therefore, we start to decrement the counter C_1 (or C_2).

Observe that $\psi_{\mathcal{M}^1}$ and $\psi_{\mathcal{M}^2}$ may still “choose” a different target label when simulating Ins_ℓ . Furthermore, even if they choose the same target label, the simulation of the counter’s updates is performed completely independently. Hence, there is no guarantee that a state t encoding a configuration (ℓ, c_1, c_2) of \mathcal{M} can reach a state encoding a successor configuration (ℓ', c'_1, c'_2) . This is enforced by the formula

$$\text{Sync} \equiv \bigwedge_{\ell=1}^m \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left((\langle r_{i, \ell}^1 \rangle_{\mathcal{A}^1} \wedge \langle r_{i, \ell}^2 \rangle_{\mathcal{A}^2}) \Rightarrow \bigvee_{\ell'=1}^m \mathbf{G}_{>0} ((\langle r_{i, \ell}^1 \rangle_{\mathcal{A}^1} \wedge \langle r_{i, \ell}^2 \rangle_{\mathcal{A}^2}) \vee (\langle r_{S(i), \ell}^1 \rangle_{\mathcal{A}^1} \wedge \langle r_{S(i), \ell'}^2 \rangle_{\mathcal{A}^2}) \vee a_{S(i), \ell'}^1) \right)$$

Technically, *Sync* says that whenever a state t satisfying $r_{i,\ell}^1 \wedge r_{i,\ell}^2$ is visited, then t has a successor t' satisfying $r_{S(i),\ell'}^1 \wedge r_{S(i),\ell'}^2$ for some ℓ' . Due to the subformulae *Struct*¹ and *Struct*², the state t' must be visited from t via a path such that all states except for t' satisfy $r_{i,\ell}^1 \wedge r_{i,\ell}^2$. Note that t' has a successor satisfying the marker $a_{S(i),\ell'}^1$. Hence, the probability of all runs initiated in t satisfying the formula

$$\mathbf{G}((r_{i,\ell}^1 \wedge r_{i,\ell}^2) \vee (r_{S(i),\ell'}^1 \wedge r_{S(i),\ell'}^2) \vee a_{S(i),\ell'}^1)$$

is positive (we could use other markers instead of $a_{S(i),\ell'}^1$).

We put

$$\varphi_{\mathcal{M}} \equiv \psi_{\mathcal{M}^1} \wedge \psi_{\mathcal{M}^2} \wedge \text{Sync} \wedge \text{Recurrent}$$

where

$$\text{Recurrent} \equiv \mathbf{G}_{=1}(\bigwedge_{\ell=1}^m (at_{\ell}^1 \wedge at_{\ell}^2) \Rightarrow \mathbf{F}_{>0}(\bigvee_{\ell' \in \tau} (at_{\ell'}^1 \wedge at_{\ell'}^2)))$$

The formula *Recurrent* enforces the existence of a run representing a τ -recurrent computation of \mathcal{M} .

We say that a run $\pi = s_0, s_1, \dots$ of a Markov chain represents a computation $\text{conf}_0, \text{conf}_1, \dots$ of \mathcal{M} if there is an infinite increasing sequence of indexes j_0, j_1, \dots such that for every $n \in \mathbb{N}$, the state s_{j_n} represents the configuration conf_n , i.e., if $\text{conf}_n = (\ell, c_1, c_2)$, then $s_{j_n} \models (at_{\ell}^1 \wedge at_{\ell}^2)$, $\gamma^1[s_{j_n}] = \text{Inc}^{c_1}(\mathbf{z})$, and $\gamma^2[s_{j_n}] = \text{Inc}^{c_2}(\mathbf{z})$.

► **Proposition 6.** *If $s \models \varphi_{\mathcal{M}}$, then there exists $\pi \in \text{Run}(s)$ representing a recurrent computation of \mathcal{M} .*

Proof. TOPROVE 1 ◀

► **Proposition 7.** *For every recurrent computation of \mathcal{M} there exist a Markov chain M , a state s of M , and $\pi \in \text{Run}(s)$ such that $s \models \varphi_{\mathcal{M}}$ and π represents the computation.*

A proof of Proposition 7 is relatively simple but technical. It can be found in Appendix.

7 The Undecidability Results

The high undecidability of PCTL satisfiability is an immediate consequence of Propositions 6 and 7. Note that the formula $\varphi_{\mathcal{M}}$ constructed in Section 6 uses only the path connective **U** and the connectives **F**, **G** definable from **U**. Hence, the high undecidability of PCTL satisfiability holds even for the **U**-fragment of PCTL. Observe that **U** is actually used only in the subformulae *NewSim* _{ℓ} ¹ and *NewSim* _{ℓ} ². In Appendix, it is shown that these formulae can be rewritten so that they use only the connectives **F** and **G**. Hence, the result holds even for the **F**, **G**-fragment of PCTL. Furthermore, when we omit the subformula enforcing the existence of a run representing a recurrent computation and assume that \mathcal{M} is deterministic, then the resulting formula has a *finite* model iff \mathcal{M} has a bounded computation. This means that *finite* PCTL satisfiability is undecidable even for the **F**, **G**-fragment of PCTL. Thus, we obtain our main result.

► **Theorem 8.** *The satisfiability problem for the **F**, **G**-fragment of PCTL is Σ_1^1 -hard, and the finite satisfiability problem for the **F**, **G**-fragment of PCTL is Σ_1^0 -hard.*

Consequently, the validity problem for the **F**, **G**-fragment is Π_1^1 -hard, and the finite validity problem for the **F**, **G**-fragment is Π_1^0 -hard. This implies that there is no complete deductive system proving all valid (or finitely valid) formulae of the **F**, **G**-fragment.

8 Conclusions

We have shown that the PCTL satisfiability problem is highly undecidable even for the **F, G**-fragment. An interesting direction for future research is to characterize the decidability border for PCTL satisfiability, i.e., to establish principle boundaries of automatic probabilistic program synthesis from PCTL specifications.

References

- 1 N. Bertrand, J. Fearnley, and S. Schewe. Bounded satisfiability for PCTL. In *Proceedings of CSL 2012*, volume 16 of *Leibniz International Proceedings in Informatics*, pages 92–106. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2012.
- 2 P. Billingsley. *Probability and Measure*. Wiley, 1995.
- 3 T. Brázdil, V. Forejt, J. Křetínský, and A. Kučera. The satisfiability problem for probabilistic CTL. In *Proceedings of LICS 2008*, pages 391–402. IEEE Computer Society Press, 2008.
- 4 J. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of STOC’88*, pages 460–467. ACM Press, 1988.
- 5 S. Chakraborty and J.P. Katoen. On the satisfiability of some simple probabilistic logics. In *Proceedings of LICS 2016*, pages 56–65, 2016.
- 6 M. Chodil and A. Kučera. The finite satisfiability problem for PCTL is undecidable. In *Proceedings of LICS 2024*, pages Article No. 22, pages 1–14. ACM Press, 2024.
- 7 M. Chodil and A. Kučera. The satisfiability problem for a quantitative fragment of PCTL. *Journal of Computer and System Sciences*, 139:103478, 2024.
- 8 A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.
- 9 S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of STOC’71*, pages 151–158. ACM Press, 1971.
- 10 E.A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science*, B:995–1072, 1991.
- 11 H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.
- 12 D. Harel. Effective transformations on infinite trees with applications to high undecidability dominoes, and fairness. *Journal of the Association for Computing Machinery*, 33(1), 1986.
- 13 S. Hart and M. Sharir. Probabilistic temporal logic for finite and bounded models. In *Proceedings of POPL’84*, pages 1–13. ACM Press, 1984.
- 14 D. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science*, 27:333–354, 1983.
- 15 D. Kozen. A finite-model theorem for the propositional μ -calculus. *Studia Logica*, 47(3):233–241, 1988.
- 16 S. Kraus and D.J. Lehmann. Decision procedures for time and chance (extended abstract). In *Proceedings of FOCS’83*, pages 202–209. IEEE Computer Society Press, 1983.
- 17 J. Křetínský and A. Rotar. The satisfiability problem for unbounded fragments of probabilistic CTL. In *Proceedings of CONCUR 2018*, volume 118 of *Leibniz International Proceedings in Informatics*, pages 32:1–32:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.
- 18 E.V. Kuzmin, V.A. Sokolov, and D.Y. Chalyy. Boundedness problems for Minsky counter machines. *Programming and Computer Software*, 36(1):3–10, 2010.
- 19 D. Lehmann and S. Shelah. Reasoning with time and chance. *Information and Control*, 53:165–198, 1982.
- 20 M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.
- 21 C. Stirling. Modal and temporal logics. *Handbook of Logic in Computer Science*, 2:477–563, 1992.

- 22 B. Trakhtenbrot. The impossibility of an algorithm for the decidability problem on finite classes. *Proceedings of the USSR Academy of Sciences*, 70(4):569–572, 1950.
- 23 A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.
- 24 I. Walukiewicz. On completeness of the μ -calculus. In *Proceedings of LICS'93*, pages 136–146. IEEE Computer Society Press, 1993.

Appendix

A Non-Deterministic d -Counter Machines

In this section, we prove Proposition 2. We start by recalling the non-deterministic Minsky machines [20] and the corresponding undecidability results.

A *non-deterministic Minsky machine* \mathcal{M} with $d \geq 1$ counters is a finite program

$$1 : Ins_1; \dots m : Ins_m;$$

where $m \geq 1$ and every $i : Ins_i$ is a labeled instruction of one of the following types:

- I. $i : inc\ c_j; goto\ L;$
- II. $i : if\ c_j=0\ then\ goto\ L\ else\ dec\ c_j; goto\ L'$

Here, $j \in \{1, \dots, d\}$ is a counter index and $L, L' \subseteq \{1, \dots, m\}$ are sets of labels with one or two elements. We say that \mathcal{M} is *deterministic* if all L, L' occurring in the instructions of \mathcal{M} are singletons¹.

A *configuration* of \mathcal{M} is a tuple (i, n_1, \dots, n_k) of non-negative integers where $1 \leq i \leq m$ represents the current control position and n_1, \dots, n_k represent the current counter values. A configuration (i', n'_1, \dots, n'_k) is a *successor* of a configuration (i, n_1, \dots, n_k) , written $(i, n_1, \dots, n_k) \mapsto (i', n'_1, \dots, n'_k)$, if the tuple (n'_1, \dots, n'_k) is obtained from (n_1, \dots, n_k) by performing Ins_i , and i' is an element of the corresponding L (or L') in Ins_i . Note that every configuration has either one or two successor(s). A *computation* of \mathcal{M} is an infinite sequence of configurations $\omega \equiv C_0, C_1, \dots$ such that $C_0 = (1, 0, \dots, 0)$ and $C_i \mapsto C_{i+1}$ for all $i \in \mathbb{N}$.

Now, we recall the standard undecidability results for Minsky machines. The symbols Σ_1^0 and Σ_1^1 denote the corresponding levels in the arithmetical and the analytical hierarchies, respectively.

(1) The *boundedness problem* for a given deterministic two-counter Minsky machine \mathcal{M} is undecidable and Σ_1^0 -complete [18]. Here, \mathcal{M} is bounded if the unique computation ω contains only finitely many pairwise different configurations.

(2) The *recurrent reachability problem* for a given non-deterministic two-counter Minsky machine \mathcal{M} is highly undecidable and Σ_1^1 -complete [12]. Here, the question is whether there exists a *recurrent* computation ω of \mathcal{M} such that the instruction Ins_1 is executed infinitely often along ω .

Observe that every instruction of a Minsky machine updates the value of just one counter and leaves the other counters unchanged. Hence, simulating Minsky machines by PCTL formulae would require inventing some mechanism for “transferring” the pairs of probabilities representing positive counter values when moving from one state to another. This is not needed when simulating our non-deterministic d -counter machines, because here every instruction updates all counters simultaneously.

► **Proposition 2.** *The problem of whether a given non-deterministic two-counter machine \mathcal{M} has a τ -recurrent computation (for a given τ) is Σ_1^1 -hard. Furthermore, the problem of whether a given deterministic two-counter machine \mathcal{M} has a bounded computation is Σ_1^0 -hard.*

Proof. TOPROVE 2

◀

¹ Our definition of non-deterministic Minsky machines is equivalent to the standard one where the target sets of labels are singletons, and there is also a Type III instruction of the form $i : goto\ u\ or\ u'$. For purposes of this paper, the adopted definition is more convenient.

B

 Representing Non-Negative Integers by Points in $(0, 1)^2$

In this section, we prove Lemma 3 and Lemma 4.

► **Lemma 3.** *Let $\lambda \in (0, \frac{1}{4})$ and $I_\lambda = \left(\frac{1-\sqrt{1-4\lambda}}{2}, \frac{1+\sqrt{1-4\lambda}}{2} \right)$. For every $\mathbf{v} \in I_\lambda \times [0, 1]$ we have the following:*

- (a) $\text{Inc}(\mathbf{v}) \in I_\lambda \times [0, 1]$;
- (b) $\text{Dec}(\text{Inc}(\mathbf{v})) = \mathbf{v}$;
- (c) $\text{Inc}(\mathbf{v})_1 < \mathbf{v}_1$ and $\text{Inc}(\mathbf{v})_2 \leq \mathbf{v}_2$; if $\mathbf{v}_2 > 0$, then $\text{Inc}(\mathbf{v})_2 < \mathbf{v}_2$;
- (d) let $\mathbf{u} = (\text{Inc}^2(\mathbf{v})_1, 0)$; then $\text{slope}(\mathbf{u}, \text{Inc}(\mathbf{v})) = \text{slope}(\text{Inc}(\mathbf{v}), \mathbf{v})$;
- (e) let $\mathbf{u} = (\text{Inc}(\mathbf{v})_1, y)$ where $0 \leq y < \text{Inc}(\mathbf{v})_2$. Then $\text{slope}(\mathbf{u}, \text{Dec}(\mathbf{u})) < \text{slope}(\text{Inc}(\mathbf{v}), \mathbf{v})$;
- (f) if \mathbf{u} belongs to the line segment between $\text{Inc}^2(\mathbf{v})$ and $\text{Inc}(\mathbf{v})$, then $\text{Dec}(\mathbf{u})$ belongs to the line segment between $\text{Inc}(\mathbf{v})$ and \mathbf{v} ;
- (g) $\lim_{n \rightarrow \infty} \text{Inc}^n(\mathbf{v})_1 = \frac{1-\sqrt{1-4\lambda}}{2}$.

Proof. TOPROVE 3 ◀

► **Lemma 4.** *Let $\lambda \in (0, \frac{1}{4})$ and $\mathbf{z} \in I_\lambda \times (0, 1)$. For all $\mathbf{v} \in I_\lambda \times [0, 1] \setminus \mathcal{A}(\mathbf{z})$ and $n \in \mathbb{N}$ such that $\text{Inc}^{n+1}(\mathbf{z})_1 \leq \mathbf{v}_1 \leq \text{Inc}^n(\mathbf{z})_1$, there exists \mathbf{u} such that $\mathbf{u}_1 = \text{Inc}^{n+1}(\mathbf{z})_1$, $0 \leq \mathbf{u}_2 < \text{Inc}^{n+1}(\mathbf{z})_2$, and \mathbf{v} belongs to the line segment between \mathbf{u} and $\text{Dec}(\mathbf{u})$.*

Proof. TOPROVE 4 ◀

C

 Avoiding the \mathbf{U} -operator

The formula $\varphi_{\mathcal{M}}$ constructed in Section 5 uses the path operator \mathbf{U} . More concretely, all occurrences of \mathbf{U} in $\varphi_{\mathcal{M}}$ are within the formula Struct . In this section, we show that using \mathbf{U} can be avoided, i.e., we design a new formula $\overline{\text{Struct}}$ such that $\overline{\text{Struct}}$ implies Struct and $\overline{\text{Struct}}$ contains only the \mathbf{F} and \mathbf{G} operators.

First, define $\mathcal{C} = \{a_{i,\ell}, \bar{a}_{i,\ell}, b_{i,\ell}, c_{i,\ell}, d_{i,\ell} \mid 0 \leq i \leq 2, 1 \leq \ell \leq m\}$. Now, define $\overline{\text{Struct}}$ as follows:

$$\overline{\text{Struct}} \equiv \overline{\text{Succ}} \wedge \text{Mark} \wedge \text{Lambda},$$

where Mark and Lambda remain as they were in the original Struct and $\overline{\text{Succ}}$ is defined as

follows:

$$\begin{aligned}
\overline{Succ} &\equiv \bigwedge_{i=0}^2 \bigwedge_{\ell=1}^m \mathbf{G}_{=1} \left(\langle r_{i,\ell} \rangle_{\mathcal{A}} \implies \bigvee_{\ell'=1}^m \mathbf{F}_{=1} rsuc_{i,\ell,\ell'} \right) \\
&\wedge \\
&\bigwedge_{i=0}^2 \bigwedge_{\ell=1}^m \mathbf{G}_{=1} \left(\langle r_{i,\ell} \rangle_{\mathcal{A}} \implies \mathbf{F}_{<1} \bigvee_{\substack{0 \leq z \leq 2 \\ 1 \leq z' \leq m \\ z \neq i \text{ or } z' \neq \ell}} \langle r_{z,z'} \rangle_{\mathcal{A}} \right) \\
&\wedge \\
&\mathbf{G}_{=1} \bigvee_{x \in \mathcal{A} \setminus \mathcal{B}} \langle x \rangle_{\mathcal{A} \setminus \mathcal{B}} \\
&\wedge \\
&\mathbf{G}_{=1} \bigwedge_{x \in \mathcal{C}} \left(x \implies \bigvee_{0 \leq j \leq 2} \langle x, R_j \rangle_{\mathcal{A}} \right) \\
&\wedge \\
&\mathbf{G}_{=1} \bigwedge_{x \in \mathcal{A} \setminus (\mathcal{B} \cup \mathcal{C})} (x \implies \langle x \rangle_{\mathcal{A}}) \\
&\wedge \\
&\bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(\langle R_i \rangle_{\mathcal{B}} \implies \mathbf{F}_{=1} Rsuc_i \right) \\
&\wedge \\
&\bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(\langle R_i, K \rangle_{\mathcal{B}} \implies \mathbf{F}_{=1} RKsuc_i \right) \\
&\wedge \\
&\bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(\left[\langle R_i \rangle_{\mathcal{B}} \vee \langle R_i, K \rangle_{\mathcal{B}} \right] \implies \mathbf{F}_{<1} \bigvee_{\substack{0 \leq z \leq 2 \\ z \neq i}} \langle R_z, K \rangle_{\mathcal{B}} \right) \\
&\wedge \\
&\mathbf{G}_{=1} \left(\left[\bigvee_{x \in \mathcal{B}} x \right] \implies \left[\mathbf{G}_{=1} \left[\bigvee_{i=0}^2 Rsuc_i \vee RKsuc_i \right] \right] \right) \\
&\wedge \\
&\mathbf{G}_{=1} (K \implies \mathbf{G}_{=1} K)
\end{aligned}$$

Clearly, \overline{Struct} is a PCTL state formula in the \mathbf{F}, \mathbf{G} fragment of PCTL. It remains to show \overline{Struct} implies $Struct$.

Suppose that \overline{Struct} holds in some state o . We want to show that $Struct$ also holds in o . Clearly $Mark$ and $Lambda$ are satisfied, so it remains to show that $Succ$ is satisfied. Recall

the definition of $Succ$:

$$\begin{aligned} Succ &\equiv \bigwedge_{\ell=1}^m \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(\langle r_{i,\ell} \rangle_{\mathcal{A}} \Rightarrow \bigvee_{\ell'=1}^m \langle r_{i,\ell} \rangle_{\mathcal{A}} \mathbf{U}_{=1} rsuc_{i,\ell,\ell'} \right) \\ &\wedge \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left((\langle R_i \rangle_{\mathcal{B}} \Rightarrow \langle R_i \rangle_{\mathcal{B}} \mathbf{U}_{=1} Rsuc_i) \wedge (\langle R_i, K \rangle_{\mathcal{B}} \Rightarrow \langle R_i, K \rangle_{\mathcal{B}} \mathbf{U}_{=1} RKsuc_i) \right) \end{aligned}$$

where

$$rsuc_{i,\ell,\ell'} \equiv \langle r_{S(i),\ell'} \rangle_{\mathcal{A}} \vee \bigvee_{j=0}^2 (\langle a_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle \bar{a}_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle b_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle c_{i,\ell}, R_j \rangle_{\mathcal{A}} \vee \langle d_{i,\ell}, R_j \rangle_{\mathcal{A}})$$

$$Rsuc_i \equiv \langle A_i \rangle_{\mathcal{B}} \vee \langle B_i \rangle_{\mathcal{B}} \vee \langle C_i \rangle_{\mathcal{B}} \vee \langle D_i \rangle_{\mathcal{B}} \vee \langle E_i \rangle_{\mathcal{B}} \vee \langle R_{S(i)}, K \rangle_{\mathcal{B}}$$

$$RKsuc_i \equiv \langle A_i, K \rangle_{\mathcal{B}} \vee \langle B_i, K \rangle_{\mathcal{B}} \vee \langle C_i, K \rangle_{\mathcal{B}} \vee \langle D_i, K \rangle_{\mathcal{B}} \vee \langle R_{S(i)}, K \rangle_{\mathcal{B}}.$$

We split the proof into two propositions to make it more readable. In the first proposition, we show that

$$\bigwedge_{\ell=1}^m \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(\langle r_{i,\ell} \rangle_{\mathcal{A}} \Rightarrow \bigvee_{\ell'=1}^m \langle r_{i,\ell} \rangle_{\mathcal{A}} \mathbf{U}_{=1} rsuc_{i,\ell,\ell'} \right)$$

holds. In the second proposition, we show that

$$\bigwedge_{i=0}^2 \mathbf{G}_{=1} \left((\langle R_i \rangle_{\mathcal{B}} \Rightarrow \langle R_i \rangle_{\mathcal{B}} \mathbf{U}_{=1} Rsuc_i) \wedge (\langle R_i, K \rangle_{\mathcal{B}} \Rightarrow \langle R_i, K \rangle_{\mathcal{B}} \mathbf{U}_{=1} RKsuc_i) \right)$$

holds.

► **Proposition 9.** *The state o satisfies the formula*

$$\bigwedge_{\ell=1}^m \bigwedge_{i=0}^2 \mathbf{G}_{=1} \left(\langle r_{i,\ell} \rangle_{\mathcal{A}} \Rightarrow \bigvee_{\ell'=1}^m \langle r_{i,\ell} \rangle_{\mathcal{A}} \mathbf{U}_{=1} rsuc_{i,\ell,\ell'} \right).$$

Proof. TOPROVE 5 ◀

► **Proposition 10.** *The state o satisfies the formula*

$$\bigwedge_{i=0}^2 \mathbf{G}_{=1} \left((\langle R_i \rangle_{\mathcal{B}} \Rightarrow \langle R_i \rangle_{\mathcal{B}} \mathbf{U}_{=1} Rsuc_i) \wedge (\langle R_i, K \rangle_{\mathcal{B}} \Rightarrow \langle R_i, K \rangle_{\mathcal{B}} \mathbf{U}_{=1} RKsuc_i) \right).$$

Proof. TOPROVE 6 ◀

D A Proof of Proposition 7

Recall that $\lambda = \frac{14}{255}$, $I_\lambda = (\frac{1}{15}, \frac{14}{15})$, $\mathbf{z} = (\frac{1}{12}, \frac{1}{15})$, and $\delta = \frac{1}{11}$. We also put $\varrho = \frac{1}{13}$. This implies

$$z_2 < \varrho < z_1 < \delta \tag{10}$$

$$2\lambda + 2\delta + 2z_1 + 2z_2 < 1 \tag{11}$$

Now we restate the proposition.

► **Proposition 7.** *For every recurrent computation of \mathcal{M} there exist a Markov chain M , a state s of M , and $\pi \in \text{Run}(s)$ such that $s \models \varphi_{\mathcal{M}}$ and π represents the computation.*

Proof. TOPROVE 7 ◀

Let us consider the formula $\Psi_{\mathcal{M}} \equiv \xi_{\mathcal{M}^1} \wedge \xi_{\mathcal{M}^2} \wedge \text{Sync}$ where $\xi_{\mathcal{M}^1}$ and $\xi_{\mathcal{M}^2}$ are obtained from $\psi_{\mathcal{M}^1}$ and $\psi_{\mathcal{M}^2}$ just by omitting the subformulae Rec^1 and Rec^2 , respectively. Then, one can easily show that a *deterministic* two-counter machine \mathcal{M} has a *bounded* computation iff $\Psi_{\mathcal{M}}$ has a finite model. The “ \Leftarrow ” direction follows by using the same arguments as in Proposition 6. The “ \Rightarrow ” direction uses a construction similar to the one of Proposition 7. Realize that if \mathcal{M} is deterministic, then a bounded run $\text{conf}_0, \text{conf}_1, \dots$ of \mathcal{M} is necessarily *periodic*, i.e., there exist $t, u \in \mathbb{N}$ such that $t < u$ and the infinite sequences $\text{conf}_t, \text{conf}_{t+1}, \dots$ and $\text{conf}_u, \text{conf}_{u+1}, \dots$ are the same. Hence, at the beginning of the construction of Proposition 7, we now add only *finitely* many states of the form $[j, r_{i,\ell}^1, r_{i,\ell}^2, c_1, c_2]$ where $j < u$. When constructing the successors of $[u-1, r_{i,\ell}^1, r_{i,\ell}^2, c_1, c_2]$, we use the state $[t, r_{S(i),\ell'}^1, r_{S(i),\ell'}^2, c'_1, c'_2]$ instead of $[u, r_{S(i),\ell'}^1, r_{S(i),\ell'}^2, c'_1, c'_2]$. Otherwise, the construction is the same. Thus, we obtain the following:

► **Corollary 11.** *The finite satisfiability problem for the **F,G**-fragment of PCTL is Σ_1^0 -hard.*