

A PROBABILISTIC HIGHER-ORDER FIXPOINT LOGIC

YO MITANI^a, NAOKI KOBAYASHI^a, AND TAKESHI TSUKADA^b

^a The University of Tokyo, Tokyo, Japan

^b Chiba University, Chiba, Japan

ABSTRACT. We introduce PHFL, a probabilistic extension of higher-order fixpoint logic, which can also be regarded as a higher-order extension of probabilistic temporal logics such as PCTL and the μ^p -calculus. We show that PHFL is strictly more expressive than the μ^p -calculus, and that the PHFL model-checking problem for finite Markov chains is undecidable even for the μ -only, order-1 fragment of PHFL. Furthermore the full PHFL is far more expressive: we give a translation from Lubarsky's μ -arithmetic to PHFL, which implies that PHFL model checking is Π_1^1 -hard and Σ_1^1 -hard. As a positive result, we characterize a decidable fragment of the PHFL model-checking problems using a novel type system.

1. INTRODUCTION

Temporal logics such as CTL and CTL* have been playing important roles in system verification. Among the most expressive temporal logics is the *higher-order fixpoint logic* (HFL for short) proposed by Viswanathan and Viswanathan [VV04], which is a higher-order extension of the *modal μ -calculus* [Koz83]. HFL is known to be strictly more expressive than the modal μ -calculus but the model-checking problem against finite models is still decidable.

In view of the increasing importance of probabilistic systems, temporal logics for probabilistic systems (such as PCTL [HJ94]) and their model-checking problems have been studied and applied to verification and analysis of probabilistic systems and randomized distributed algorithms [KNP11]. Recently Castro et al. [CKP15] have proposed a probabilistic extension of the modal μ -calculus, called the *μ^p -calculus*. They showed that the μ^p -calculus is strictly more expressive than PCTL and that the model-checking problem for the μ^p -calculus belongs to $\text{NP} \cap \text{co-NP}$.

In the present paper, we introduce PHFL, a probabilistic higher-order fixpoint logic, and studies the model checking problem. PHFL can be regarded as a probabilistic extension of HFL and as a higher-order extension of the μ^p -calculus. PHFL strictly subsumes the μ^p -calculus [CKP15], which coincides with order-0 PHFL.

We prove that PHFL model checking for finite Markov chains is undecidable even for the order-1 fragment of PHFL without fixpoint alternations, by giving a reduction from the *value problem* of probabilistic automata [Rab63, Paz71]. In the presence of fixpoint alternations (i.e., with both least and greatest fixpoint operators), PHFL model checking

Key words and phrases: Probabilistic logics, higher-order fixpoint logic, model checking.

is even harder: the order-1 PHFL model-checking problem is Π_1^1 -hard and Σ_1^1 -hard. The proof is by a reduction from the validity checking problem for μ -arithmetic [Lub89] to PHFL model checking. This may be surprising, because both order-0 PHFL model checking (i.e. μ^p -calculus model checking) for finite Markov chains [CKP15] and HFL model checking for finite state systems [VV04] are decidable. The combination of probabilities and higher-order predicates suddenly makes the model-checking problem highly undecidable.

As a positive result, we identify a decidable subclass of PHFL model-checking problems. To characterize the subclass, we introduce a type system for PHFL formulas, which is parameterized by a Markov chain M . We show that the model-checking problem $M \models \varphi$ is decidable provided that φ is typable in the type system for M , by giving a decision procedure using the decidability of existential theories of reals. The decidable subclass is reasonably expressive: the problem of computing termination probabilities of *recursive Markov chains* [EY09] can be reduced to the subclass.

The rest of this article is organized as follows. Section 2 introduces PHFL and shows that it is strictly more expressive than the μ^p -calculus. Section 3 proves undecidability of the model-checking problem for μ -only and order-1 PHFL. Section 4 proves that the PHFL model-checking problem is both Π_1^1 -hard and Σ_1^1 -hard. Section 5 introduces a decidable subclass of PHFL model-checking problems, and shows that the subclass is reasonably large. Section 6 discusses related work, and Section 7 concludes the paper. A preliminary summary of this article has been published in Proceedings of FSCD 2020 [MKT20]. This article contains details omitted in the preliminary summary, and also significantly extends the decidable fragment of PHFL in Section 5.

2. PHFL: PROBABILISTIC HIGHER-ORDER FIXPOINT LOGIC

This section introduces PHFL, a probabilistic extension of HFL [VV04]. PHFL is a logic used for describing properties of Markov chains. We define its syntax and semantics and show that it is more expressive than the μ^p -calculus [CKP15].

2.1. Markov Chains. We first recall the standard notion of Markov chains. Our definitions follow those in [CKP15].

Definition 2.1. Let AP be a set of atomic propositions. A *Markov chain* over AP is a tuple $(S, P, \rho_{AP}, s_{\text{in}})$, where:

- S is a finite set of states,
- $P : S \times S \rightarrow [0, 1]$ satisfying $\sum_{s' \in S} P(s, s') = 1$ for every $s \in S$, describes transition probabilities,
- $\rho_{AP} : AP \rightarrow 2^S$ is a labeling function, and
- $s_{\text{in}} \in S$ is an initial state.

For a Markov chain $M = (S, P, \rho_{AP}, s_{\text{in}})$, its *embedded Kripke structure* is $K = (S, R, \rho_{AP}, s_{\text{in}})$ where $R \subseteq S \times S$ is a relation such that $R = \{(s, s') \mid P(s, s') > 0\}$.

Intuitively, $P(s, s')$ denotes the probability that the state s transits to the state s' , and $\rho_{AP}(p)$ gives the set of states where p is true. Throughout the paper, we assume that the set AP of atomic propositions is closed under negations, in the sense that for any $p \in AP$, there exists $\bar{p} \in AP$ such that $\rho_{AP}(\bar{p}) = S \setminus \rho_{AP}(p)$.

Given a Markov chain M , we often write $S_M, P_M, \rho_{AP, M}, s_{\text{in}, M}$ for its components; we omit the subscript M when it is clear from the context.

2.2. Syntax of PHFL Formulas. As in HFL [VV04, KLB17], we need the notion of types to define the syntax of PHFL formulas.

The set of types, ranged over by τ , is given by:

$$\tau ::= Prop \mid \tau_1 \rightarrow \tau_2.$$

The type $Prop$ describes *quantitative* propositions, whose values range over $[0, 1]$. Intuitively, the value of a quantitative proposition represents the *probability* that the proposition holds. The type $\tau_1 \rightarrow \tau_2$ is for functions from τ_1 to τ_2 . For example, $(Prop \rightarrow Prop) \rightarrow Prop$ represents the type of (higher-order, quantitative) predicates on unary predicates.

Remark 2.2. In the μ^p -calculus [CKP15] and the previous version of this paper [MKT20], two kinds of propositions were considered: *quantitative* propositions, which take values in $[0, 1]$, and *qualitative* propositions, which take truth values. In the present paper, we consider only quantitative propositions for the sake of simplicity, and regard qualitative propositions as a special case of the former by treating 0 and 1 as the truth values “false” and “true” respectively.

We assume a countably infinite set Var of variables, ranged over by X_1, X_2, \dots . The set of PHFL (pre-)formulas, ranged over by ϕ , is given by:

$$\phi ::= p \mid X \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid [\phi]_J \mid \Box\phi \mid \Diamond\phi \mid \bigcirc\phi \mid \mu X.\phi \mid \nu X.\phi \mid \lambda X.\phi \mid \phi_1 \phi_2.$$

Here, p ranges over the set AP of atomic propositions (of the underlying Markov chains; we thus assume that AP is closed under negations). The subscript J of $[\phi]_J$ is either “ $> r$ ” or “ $\geq r$ ” for some rational number $r \in [0, 1]$. We often identify J with an interval: for example, “ $> r$ ” is regarded as $(r, 1] = \{x \mid r < x \leq 1\}$. Given a quantitative proposition ϕ , the formula $[\phi]_{>r}$ (resp. $[\phi]_{\geq r}$) is a qualitative formula, which is true just if the probability that ϕ holds is greater than r (resp. no less than r). We exclude trivial bounds “ > 1 ” and “ ≥ 0 ”; note that $[\phi]_{>1}$ and $[\phi]_{\geq 0}$ are equivalent to false and true respectively. The formulas $\Box\phi$, $\Diamond\phi$, and $\bigcirc\phi$ respectively mean the minimum, maximum, and average probabilities that ϕ holds after a one-step transition. The formulas $\mu X.\phi$ and $\nu X.\phi$ respectively denote the least and greatest fixpoints of $\lambda X.\phi$. Note that ϕ may denote higher-order predicates, as in HFL [VV04] (but unlike in the modal μ -calculus and its probabilistic variants [CKP15, MS13, MM97], where fixpoints are restricted to propositions). We have also λ -abstractions and applications, which are used for manipulating higher-order predicates. The prefixes μX , νX and λX bind the variable X . As usual, we identify formulas up to the renaming of bound variables and implicitly apply α -conversions. We write $[\phi_1/X]$ for the capture-avoiding substitution of ϕ_1 for X , and $[\phi_1/X]\phi_2$ for the formula obtained by applying the substitution $[\phi_1/X]$ to ϕ_2 .

In order to exclude out ill-formed formulas like $(p_1 \vee p_2)(\phi)$, we restrict the shape of formulas using a simple type system. A *type environment* is a map from a finite set of variables to the set of types. A *type judgment* is of the form $\Gamma \vdash \phi : \tau$. The typing rules are shown in Figure 1. A formula ϕ is *well-typed* if $\Gamma \vdash \phi : \tau$ is derivable for some Γ and τ . Henceforth, we consider only well-typed formulas.

Example 2.3. For a proposition $p \in AP$, the formula $\phi = (\mu F.\lambda X.X \vee F(\bigcirc X))p$ is a well-typed formula of type $Prop$. By unfolding the fixpoint formula (i.e., replacing $\mu X.\phi$ with $[\mu X.\phi/X]\phi$, which will be justified by the semantics introduced later) and applying β -reductions, we obtain:

$$\begin{aligned} \phi &\equiv (\lambda X.X \vee (\mu F.\lambda X.X \vee F(\bigcirc X))(\bigcirc X))p \\ &\equiv p \vee (\mu F.\lambda X.X \vee F(\bigcirc X))(\bigcirc p) \end{aligned}$$

$$\begin{array}{c}
\frac{}{\Gamma \vdash p : Prop} \quad \frac{}{\Gamma, X : \tau \vdash X : \tau} \quad \frac{\Gamma \vdash \phi : Prop}{\Gamma \vdash [\phi]_J : Prop} \\
\\
\frac{\Gamma \vdash \phi_1 : Prop \quad \Gamma \vdash \phi_2 : Prop}{\Gamma \vdash \phi_1 \vee \phi_2 : Prop} \quad \frac{\Gamma \vdash \phi_1 : Prop \quad \Gamma \vdash \phi_2 : Prop}{\Gamma \vdash \phi_1 \wedge \phi_2 : Prop} \\
\\
\frac{\Gamma \vdash \phi : Prop}{\Gamma \vdash \Box \phi : Prop} \quad \frac{\Gamma \vdash \phi : Prop}{\Gamma \vdash \Diamond \phi : Prop} \quad \frac{\Gamma \vdash \phi : Prop}{\Gamma \vdash \bigcirc \phi : Prop} \quad \frac{\Gamma, X : \tau \vdash \phi : \tau}{\Gamma \vdash \mu X. \phi : \tau} \\
\\
\frac{\Gamma, X : \tau \vdash \phi : \tau}{\Gamma \vdash \nu X. \phi : \tau} \quad \frac{\Gamma, X : \tau_1 \vdash \phi : \tau_2}{\Gamma \vdash \lambda X. \phi : \tau_1 \rightarrow \tau_2} \quad \frac{\Gamma \vdash \phi : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash \psi : \tau_1}{\Gamma \vdash \phi \psi : \tau_2}
\end{array}$$

Figure 1: Typing rules for PHFL formulas.

$$\begin{aligned}
&\equiv p \vee \bigcirc p \vee (\mu F. \lambda X. X \vee F(\bigcirc X))(\bigcirc \bigcirc p) \\
&\equiv p \vee \bigcirc p \vee \bigcirc \bigcirc p \vee \dots
\end{aligned}$$

Thus, intuitively, the formula represents the function that maps the current state s to the value $\sup_{k \geq 0} q_k$ where q_k is the probability that a k -step transition sequence starting from the state s ends in a state satisfying p . \square

Remark 2.4. Following the definition of HFL by Kobayashi et al. [KLB17], we have excluded out negations. By a transformation similar to that in [Loz15] and our assumption that the set of atomic propositions is closed under negations, any closed ground-type formula of PHFL extended with negations can be transformed to an equivalent negation-free formula (as long as the occurrences of negations are restricted as in the original HFL [VV04] so that fixpoint operators are applied to only monotonic functions). \square

We define the *order* of a type τ by:

$$order(Prop) = 0 \quad order(\tau_1 \rightarrow \tau_2) = \max(order(\tau_1) + 1, order(\tau_2)).$$

The order of a formula ϕ such that $\Gamma \vdash \phi : \tau$ is the largest order of types used in the derivation of $\Gamma \vdash \phi : \tau$. The *order- k PHFL* is the fragment of PHFL consisting of formulas of order up to k . Order-0 PHFL coincides with the μ^p -calculus [CKP15].

2.3. Semantics. We first give the semantics of types. We write $\leq_{\mathbb{R}}$ for the natural order over the set \mathbb{R} of real numbers, and often omit the subscript when there is no danger of confusion. For a map f , we write $dom(f)$ for the domain of f .

Definition 2.5 (Semantics of Types). Let M be a Markov chain. For each τ , we define a partially ordered set $\llbracket \tau \rrbracket_M = (D_\tau, \sqsubseteq_\tau)$ inductively by:

$$\begin{aligned}
D_{Prop} &= S_M \rightarrow [0, 1] & f \sqsubseteq_{Prop} g &\stackrel{\text{def}}{\iff} \forall s \in S_M. f(s) \leq g(s) \\
D_{\tau_1 \rightarrow \tau_2} &= \{f \in D_{\tau_1} \rightarrow D_{\tau_2} \mid \forall x, y \in D_{\tau_1}. x \sqsubseteq_{\tau_1} y \implies f(x) \sqsubseteq_{\tau_2} f(y)\} \\
f \sqsubseteq_{\tau_1 \rightarrow \tau_2} g &\stackrel{\text{def}}{\iff} \forall x \in D_{\tau_1}. f(x) \sqsubseteq_{\tau_2} g(x).
\end{aligned}$$

For a type environment Γ , we write $\llbracket \Gamma \rrbracket_M$ for the set of maps f such that $dom(f) = dom(\Gamma)$ and $f(x) \in D_{\Gamma(x)}$ for every $x \in dom(\Gamma)$.

We omit the subscript M below. Note that $[[\tau]]$ forms a complete lattice for each τ . We write \perp_τ for the least element of $[[\tau]]$, and for a set $V \subseteq D_\tau$, we write $\bigsqcup_\tau V$ ($\bigsqcap_\tau V$, resp.) for the least upper bound (greatest lower bound, resp.) of V with respect to \sqsubseteq_τ ; we often omit the subscript τ if it is clear from the context. Note also that for every functional type $\tau_1 \rightarrow \tau_2$, every element of $D_{\tau_1 \rightarrow \tau_2}$ is monotonic. Thus, for every type τ and every function $f \in D_{\tau \rightarrow \tau}$, there exist the least and greatest fixed points of f , which we write $LFP(f)$ and $GFP(f)$ respectively. They are given by:

$$LFP(f) = \bigsqcap_\tau \{x \in D_\tau \mid f x \sqsubseteq_\tau x\} \quad GFP(f) = \bigsqcup_\tau \{x \in D_\tau \mid x \sqsubseteq_\tau f x\}.$$

We now define the semantics of formulas. Since the meaning of a formula depends on its type environment, we actually define the semantics $[[\Gamma \vdash \phi : \tau]]_M$ for each type judgment $\Gamma \vdash \phi : \tau$. Here, the subscript M denotes the underlying Markov chain, which is often omitted.

Definition 2.6 (Semantics of Type Judgement). Let M be a Markov chain and assume that $\Gamma \vdash \phi : \tau$ is derivable. Then its semantics $[[\Gamma \vdash \phi : \tau]]_M \in [[\Gamma]] \rightarrow [[\tau]]$ is defined by induction on the (unique) derivation of $\Gamma \vdash \phi : \tau$ by:

$$\begin{aligned} [[\Gamma \vdash p : Prop]]_M(\rho) &= \lambda s \in S_M. \text{if } s \in \rho_{AP,M}(p) \text{ then } 1 \text{ else } 0 \\ [[\Gamma \vdash X : \tau]]_M(\rho) &= \rho(X) \\ [[\Gamma \vdash \phi_1 \wedge \phi_2 : Prop]]_M(\rho) &= \lambda s \in S_M. \min_{i \in \{1,2\}} [[\Gamma \vdash \phi_i : Prop]]_M(\rho)(s) \\ [[\Gamma \vdash \phi_1 \vee \phi_2 : Prop]]_M(\rho) &= \lambda s \in S_M. \max_{i \in \{1,2\}} [[\Gamma \vdash \phi_i : Prop]]_M(\rho)(s) \\ [[\Gamma \vdash [\phi]_J : Prop]]_M(\rho) &= \lambda s \in S_M. \text{if } [[\Gamma \vdash \phi : Prop]]_M(\rho)(s) \in J \text{ then } 1 \text{ else } 0 \\ [[\Gamma \vdash \Box \phi : Prop]]_M(\rho) &= \lambda s \in S_M. \min_{s': P_M(s,s') > 0} [[\Gamma \vdash \phi : Prop]]_M(\rho)(s') \\ [[\Gamma \vdash \Diamond \phi : Prop]]_M(\rho) &= \lambda s \in S_M. \max_{s': P_M(s,s') > 0} [[\Gamma \vdash \phi : Prop]]_M(\rho)(s') \\ [[\Gamma \vdash \bigcirc \phi : Prop]]_M(\rho) &= \lambda s \in S_M. \sum_{s' \in S_M} (P_M(s, s') \cdot [[\Gamma \vdash \phi : Prop]]_M(\rho)(s')) \\ [[\Gamma \vdash \mu X. \phi : \tau]]_M(\rho) &= LFP(\lambda v \in D_\tau. [[\Gamma, X : \tau \vdash \phi : \tau]]_M(\rho[X \mapsto v])) \\ [[\Gamma \vdash \nu X. \phi : \tau]]_M(\rho) &= GFP(\lambda v \in D_\tau. [[\Gamma, X : \tau \vdash \phi : \tau]]_M(\rho[X \mapsto v])) \\ [[\Gamma \vdash \lambda X. \phi : \tau_1 \rightarrow \tau_2]]_M(\rho) &= \lambda v \in D_{\tau_1}. [[\Gamma, X : \tau_1 \vdash \phi : \tau_2]]_M(\rho[X \mapsto v]) \\ [[\Gamma \vdash \phi_1 \phi_2 : \tau]]_M(\rho) &= [[\Gamma \vdash \phi_1 : \tau_2 \rightarrow \tau]]_M(\rho) ([[\Gamma \vdash \phi_2 : \tau_2]]_M(\rho)) \end{aligned}$$

In the last equality, τ_2 is uniquely determined from Γ and ϕ_2 . In the definitions of the semantics of $\Box \phi$ and $\Diamond \phi$, the set $\{s' \in S_M \mid P(s, s') > 0\}$ is non-empty and finite, because $\sum_{s' \in S_M} P(s, s') = 1$ and S_M is finite by the definition of Markov chains. Thus the max/min operations are well-defined. We also note that $[[\Gamma \vdash \phi : \tau]]$ is a monotone function from $[[\Gamma]]$ to $[[\tau]]$ (where $[[\Gamma]]$ is ordered by the component-wise ordering; note also Remark 2.7 below). This ensures the well-definedness of the semantics of $\mu X. \phi$, $\nu X. \phi$, and $\lambda X. \phi$.

Remark 2.7. Recall that in a formula $[\phi]_J$, we allow the predicate J to be “ $> r$ ” or “ $\geq r$ ” (where $r \in [0, 1]$), but neither “ $< r$ ” nor “ $\leq r$ ”. Allowing “ $< r$ ” would break the monotonicity of the semantics of a formula. For example, $[[\emptyset \vdash \lambda X. [X]_{<1} : Prop \rightarrow Prop]] = \lambda v \in D_{Prop}. \lambda s \in S. (\text{if } v(s) < 1 \text{ then } 1 \text{ else } 0)$ is not monotonic. \square

We often omit M , the type of the formula, and the type environment, and just write $\llbracket \phi \rrbracket$ or $\llbracket \Gamma \vdash \phi \rrbracket$ for $\llbracket \Gamma \vdash \phi : \tau \rrbracket_M$ when there is no danger of confusion. For a Markov chain $M = (S, P, \rho_{AP}, s_{\text{in}})$ and a closed PHFL formula ϕ of type $Prop$, we write $M \models \phi$ if $\llbracket \phi \rrbracket(s_{\text{in}}) = 1$.

Example 2.8. Recall the PHFL formula $\phi = \psi p$ where $\psi = \mu F.\lambda X.X \vee F(\bigcirc X)$ in Example 2.3. We have:

$$\begin{aligned} \llbracket \psi \rrbracket &= LFP \left(\lambda v \in D_{Prop \rightarrow Prop} . \lambda x \in D_{Prop} . \lambda s \in S . \right. \\ &\quad \left. \max \left(x s, v (\lambda s' \in S . \sum_{s''} P(s', s'') \cdot (x s'')) s \right) \right) \\ &\geq \left(\lambda v . \lambda x . \lambda s . \max \left(x s, v (\lambda s' \in S . \sum_{s''} P(s', s'') \cdot (x s'')) s \right) \right)^{n+1} (\perp_{Prop \rightarrow Prop}) \\ &= \lambda x . \lambda s . \max_{0 \leq k \leq n} \sum_{s_0 s_1 \dots s_k \in S^{k+1}, s_0 = s} (x(s_k) \cdot \prod_{0 \leq j \leq k-1} P(s_j, s_{j+1})) \end{aligned}$$

for every $n \geq 0$. Thus, we have:

$$\llbracket \psi \rrbracket \geq \lambda x . \lambda s \in S . \sup_{k \geq 0} \sum_{s_0 s_1 \dots s_k \in S^{k+1}, s_0 = s} (x(s_k) \cdot \prod_{0 \leq j \leq k-1} P(s_j, s_{j+1})).$$

Actually, the equality holds, because the righthand side is a fixpoint of

$$\lambda v \in D_{Prop \rightarrow Prop} . \lambda x \in D_{Prop} . \lambda s \in S . \max(x s, v (\lambda s' \in S . \sum_{s''} P(s', s'') \cdot (x s''))).$$

The semantics of ϕ is, therefore, given by

$$\llbracket \phi \rrbracket = \lambda s \in S . \sup_{k \geq 0} \sum_{s_0 s_1 \dots s_k \in S^{k+1}, s_0 = s, s_k \in \rho_{AP}(p)} \prod_{0 \leq j \leq k-1} P(s_j, s_{j+1}). \quad \square$$

2.4. Expressive Power. PHFL obviously subsumes the μ^p -calculus [CKP15], which coincides with order-0 PHFL. Hence PHFL also subsumes PCTL [HJ94], since the μ^p -calculus subsumes PCTL [CKP15].

PHFL is *strictly* more expressive than the μ^p -calculus.

Theorem 2.9. *Order-1 PHFL is strictly more expressive than the μ^p -calculus, i.e., there exists an order-1 PHFL proposition ϕ such that ϕ is not equivalent to any μ^p -formula.*

Proof. Let \mathcal{M} be the set of Markov chains $M = (S, P, \rho_{AP}, s_{\text{in}})$ that satisfy the following conditions.

- $S = \{s_0, s_1, \dots, s_n\}$ for a positive integer n ,
- $P(s_i, s_{i+1}) = 1$ ($0 \leq i \leq n-1$), $P(s_n, s_n) = 1$ and $P(s_i, s_j) = 0$ otherwise.
- There are three atomic propositions a, b, c with $\rho_{AP}(a) \cup \rho_{AP}(b) = \{s_0, s_1, \dots, s_{n-1}\}$, $\rho_{AP}(a) \cap \rho_{AP}(b) = \emptyset$ and $\rho_{AP}(c) = \{s_n\}$.
- The initial state is $s_{\text{in}} = s_0$.

Let ϕ be the order-1 PHFL formula of type $Prop$:

$$(\mu F.\lambda X.a \wedge \diamond(X \vee F(b \wedge \diamond X)))(b \wedge \diamond c).$$

Note that, for $M \in \mathcal{M}$, $M \models \phi$ holds just if n is even, $\rho_{AP}(a) = \{s_0, s_1, \dots, s_{\frac{n}{2}-1}\}$ and $\rho_{AP}(b) = \{s_{\frac{n}{2}}, s_{\frac{n}{2}+1}, \dots, s_{n-1}\}$.

We show that there is no μ^p -formula equivalent to ϕ . Suppose that a μ^p -formula ϕ' were equivalent to ϕ , which would imply that $M \models \phi$ if and only if $M \models \phi'$ for any $M \in \mathcal{M}$. For $M \in \mathcal{M}$, let us write K_M for the embedded Kripke structure of M . Since all the transitions in M are deterministic, there exists a modal μ -calculus formula ϕ'' such that $M \models \phi'$ if and only if $K_M \models \phi''$ (note that ϕ'' is obtained by replacing \bigcirc with \diamond , and replacing $[\phi_1]_J$ with true if J is “ ≥ 0 ” and with ϕ_1 otherwise). That would imply that $K_M \models \phi''$ for $M \in \mathcal{M}$, just if n is even and ρ_{AP} satisfies $\rho_{AP}(a) = \{s_0, s_1, \dots, s_{\frac{n}{2}-1}\}$ and $\rho_{AP}(b) = \{s_{\frac{n}{2}}, s_{\frac{n}{2}+1}, \dots, s_{n-1}\}$. But then ϕ'' would describe the non-regular language $\{a^m b^m \mid m \geq 1\}$, which contradicts the fact that the modal μ -calculus can express only regular properties. \square

Remark 2.10. For non-probabilistic logics, HFL was known to be strictly more expressive than the modal μ -calculus [VV04]. The above proof can be easily adapted to show that fact.

3. UNDECIDABILITY OF PHFL MODEL CHECKING

In this section we prove the undecidability of the following problem.

Definition 3.1 (PHFL Model Checking). The *PHFL model-checking problem* for finite Markov chains is the problem of deciding whether $M \models \phi$, given a (finite) Markov chain M and a closed PHFL formula ϕ of type *Prop* as input.

We prove that the problem is undecidable even for the order-1 fragment of PHFL without fixpoint alternations, by a reduction from the undecidability of the value-1 problem [GO10] for probabilistic automata [Rab63]. In contrast to the undecidability of PHFL model checking, the corresponding model-checking problems are *decidable* for the full fragments of the μ^p -calculus [CKP15] and (non-probabilistic) HFL [VV04], with fixpoint alternations. Thus, the combination of probabilities and higher-order predicates introduces a new difficulty.

In Section 3.1, we review the definition of probabilistic automata and the value-1 problem. Section 3.2 shows the reduction from the value-1 problem to the PHFL model-checking problem.

3.1. Probabilistic Automata. We review probabilistic automata [Rab63] and the undecidability of the value-1 problem. Our definition follows [Fij17].

Definition 3.2 (Probabilistic Automata). A *probabilistic automaton* A is a quintuple $(Q, \Sigma, q_I, \Delta, F)$ where

- Q is a finite set of states,
- Σ is a finite set of input symbols,
- $q_I \in Q$ is an initial state,
- $\Delta : Q \times \Sigma \rightarrow D(Q)$, where $D(Q) := \{f : Q \rightarrow [0, 1] \mid \sum_{q \in Q} f(q) = 1\}$ is the set of probabilistic distributions over the set Q , represents transition probabilities, and
- $F \subseteq Q$ is a set of accepting states.

For a word $w = w_1 \cdots w_n \in \Sigma^n$, the probability that w is accepted by $A = (Q, \Sigma, q_I, \Delta, F)$, written $A(w)$, is defined by:

$$A(w) := \sum_{\substack{q_0, \dots, q_{n-1} \in Q, q_n \in F \\ \text{s.t. } q_0 = q_I}} \prod_{1 \leq i \leq n} \Delta(q_{i-1}, w_i)(q_i).$$

The *value* of a probabilistic automaton A , denoted by $\text{val}(A)$, is defined by

$$\text{val}(A) := \sup_{w \in \Sigma^*} A(w).$$

The problem of deciding whether $\text{val}(A) = 1$, called the *value-1 problem*, is known to be undecidable.

Theorem 3.3 (Undecidability of The Value-1 Problem [GO10]). *Given a probabilistic automaton A , whether $\text{val}(A) = 1$ is undecidable.*

3.2. The Undecidability Result. Let $A = (Q, \Sigma, q_I, \Delta, F)$ be a probabilistic automaton, where $\Sigma = \{c_1, \dots, c_{|\Sigma|}\}$ with $|\Sigma| > 0$. We shall construct a Markov chain M_A and a PHFL formula ϕ_A , so that $\text{val}(A) = 1$ if and only if $M_A \models \phi_A$. The undecidability of PHFL model checking then follows immediately from Theorem 3.3.

We first construct the Markov chain M_A . The set AP of atomic propositions is $\{p_c \mid c \in \Sigma\} \uplus \{p_F\}$. The Markov chain $M_A = (S, P, \rho_{AP}, s_{\text{in}})$ is defined as follows.

- The set S of states is $Q \uplus (Q \times \Sigma)$.
- The transition probability P is given by:

$$\begin{aligned} P((q, c), q') &= \Delta(q, c)(q') && (c \in \Sigma \text{ and } q, q' \in Q) \\ P(q, (q, c)) &= \frac{1}{|\Sigma|} && (c \in \Sigma \text{ and } q \in Q) \\ P(s, s') &= 0 && (\text{otherwise}) \end{aligned}$$

The first transition (from (q, c) to q') is used to simulate the transition of A from q to q' for the input symbol c . The second transition (from q to (q, c)) is used to choose the next input symbol to be supplied to the automaton; the probability is not important, and replacing $1/|\Sigma|$ with any non-zero probability does not affect the arguments below.

- ρ_{AP} is defined by:

$$\rho_{AP}(p_c) = \{(q, c) \mid q \in Q\} \quad \rho_{AP}(p_F) = \{q \mid q \in F\}.$$

- The initial state is $s_{\text{in}} = q_I$.

Intuitively, the Markov chain M_A simulates the behavior of A . The atomic proposition p_c means that A is currently reading the symbol c , and p_F means that A is in a final state.

Based on this intuition, we now construct the PHFL formula ϕ_A . For each $c \in \Sigma$, we define a formula f_c of type $Prop \rightarrow Prop$ by:

$$f_c := \lambda X. \diamond(p_c \wedge \bigcirc X).$$

Intuitively $f_c(\phi)$ denotes the probability that the automaton transits to a state satisfying ϕ given c as the next input. Given a word $w = w_1 w_2 \dots w_n \in \Sigma^*$, we define the formula g_w by

$$g_w := f_{w_1}(f_{w_2}(\dots(f_{w_n} p_F) \dots)).$$

We write A_q for the automaton obtained from A by replacing the initial state with q . The following lemma states that g_w represents the probability that w is accepted by the automaton from the current state q .

Lemma 3.4. $A_q(w) = \llbracket g_w \rrbracket_{M_A}(q)$ for every $q \in Q$.

Proof. Let $A = (Q, \Sigma, q_I, \Delta, F)$. The proof proceeds by induction on the length $|w|$ of w .

- Case where $|w| = 0$, i.e., $w = \epsilon$: By the definition of A_q , $A_q(\epsilon) = 1$ if $q \in F$ and 0 otherwise. We have the required result, as $g_\epsilon = p_F$.

- Case where $|w| > 0$: Let $w = w_1 \cdots w_n = w_1 w'$. We have:

$$\begin{aligned} A_q(w) &= \sum_{q_0, \dots, q_{n-1} \in Q, q_n \in F \text{ s.t. } q_0 = q} \prod_{1 \leq i \leq n} \Delta(q_{i-1}, w_i)(q_i) \\ &= \sum_{q' \in Q} \Delta(q, w_1)(q') \cdot \left(\sum_{q_1, \dots, q_{n-1} \in Q, q_n \in F \text{ s.t. } q_1 = q'} \prod_{2 \leq i \leq n} \Delta(q_{i-1}, w_i)(q_i) \right) \\ &= \sum_{q' \in Q} \Delta(q, w_1)(q') \cdot A_{q'}(w'). \end{aligned}$$

Since $g_w = f_{w_1}(g_{w'}) \equiv \diamond(p_{w_1} \wedge \bigcirc g_{w'})$, we have:

$$\begin{aligned} \llbracket g_w \rrbracket_{M_A}(q) &= \max_{c \in \Sigma} \llbracket p_{w_1} \wedge \bigcirc g_{w'} \rrbracket(q, c) \\ &= \llbracket \bigcirc g_{w'} \rrbracket(q, w_1) \\ &= \sum_{q' \in Q} \Delta(q, w_1)(q') \cdot \llbracket g_{w'} \rrbracket(q') \end{aligned}$$

By the induction hypothesis, we have $A_{q'}(w') = \llbracket g_{w'} \rrbracket(q')$, which implies the the required result. \square

Using Lemma 3.4, we obtain $\text{val}(A) = \sup_{n \in \omega} \llbracket \bigvee_{w \in \Sigma^{\leq n}} g_w \rrbracket_{M_A}(q_I)$, where $\Sigma^{\leq n}$ is the set of words of length up to n . This can be expressed by using the least fixpoint operator.

Theorem 3.5. Let θ_A be the formula of type $\text{Prop} \rightarrow \text{Prop}$ defined by:

$$\theta_A := \mu F. (\lambda X. X \vee \bigvee_{c \in \Sigma} F(f_c X)).$$

Then $\text{val}(A) = \llbracket \theta_A p_F \rrbracket_{M_A}(q_I)$. Therefore $M_A \models \phi_A$ if and only if $\text{val}(A) = 1$, for $\phi_A := \llbracket \theta_A p_F \rrbracket_{\geq 1}$.

Proof. Let

$$\xi := \lambda F. \lambda X. X \vee \bigvee_{c \in \Sigma} F(f_c X).$$

Then, we have

$$\llbracket \theta_A \rrbracket_M = \llbracket \mu F. \xi F \rrbracket_M = \bigsqcup_{\text{Prop} \rightarrow \text{Prop}} \{ \llbracket \xi^n(\perp) \rrbracket \mid n \in \omega \}$$

where $\perp := \lambda Z. \mu U. U$ is the formula of type $\text{Prop} \rightarrow \text{Prop}$, and $\xi^n(x)$ denotes n -times applications of ξ to x . In fact, $\bigsqcup_{\text{Prop} \rightarrow \text{Prop}} \{ \llbracket \xi^n(\perp) \rrbracket \mid n \in \omega \}$ is a fixpoint of $\llbracket \xi \rrbracket$, because:

$$\begin{aligned} &\llbracket \xi \rrbracket(\bigsqcup_{\text{Prop} \rightarrow \text{Prop}} \{ \llbracket \xi^n(\perp) \rrbracket \mid n \in \omega \}) \\ &= \lambda x \in D_{\text{Prop} \cdot x} \bigsqcup_{\text{Prop}} (\bigsqcup_{c \in \Sigma} (\bigsqcup_{\text{Prop} \rightarrow \text{Prop}} \{ \llbracket \xi^n(\perp) \rrbracket \mid n \in \omega \}))(\llbracket f_c \rrbracket x) \end{aligned}$$

$$\begin{aligned}
&= \lambda x \in D_{Prop} \cdot x \sqcup_{Prop} \left(\bigsqcup_{c \in \Sigma} \left(\bigsqcup_{Prop} \{ \llbracket \xi^n(\perp) \rrbracket \} (\llbracket f_c \rrbracket x) \mid n \in \omega \} \right) \right) \\
&= \lambda x \in D_{Prop} \cdot \bigsqcup_{Prop} \{ x \sqcup_{Prop} \left(\bigsqcup_{c \in \Sigma} \llbracket \xi^n(\perp) \rrbracket \} (\llbracket f_c \rrbracket x) \mid n \in \omega \} \right) \\
&= \bigsqcup_{Prop \rightarrow Prop} \{ \llbracket \xi^{n+1}(\perp) \rrbracket \mid n \in \omega \}.
\end{aligned}$$

Since $\llbracket \xi \rrbracket$ is monotonic and $\llbracket \perp \rrbracket$ is the least element, we also have:

$$\llbracket \mu F. \xi F \rrbracket_M = \llbracket \xi^n(\mu F. \xi F) \rrbracket \supseteq \llbracket \xi^n(\perp) \rrbracket$$

for any $n \in \omega$ hence also

$$\llbracket \mu F. \xi F \rrbracket_M \supseteq \bigsqcup_{Prop \rightarrow Prop} \{ \llbracket \xi^n(\perp) \rrbracket \mid n \in \omega \}.$$

Thus, we have the equality.

By a straightforward induction on n , we also have: $\llbracket \xi^{n+1}(\perp) p_F \rrbracket_M = \llbracket \bigvee_{w \in \Sigma^{\leq n}} g_w \rrbracket_M$. Therefore, by using also Lemma 3.4, we obtain:

$$\text{val}(A) = \sup_n \left(\llbracket \bigvee_{w \in \Sigma^{\leq n}} g_w \rrbracket_{M_A}(q_I) \right) = \sup_n \left(\llbracket \xi^{n+1}(\perp) p_F \rrbracket(q_I) \right) = \llbracket \theta_A p_F \rrbracket_{M_A}(q_I),$$

which implies the required result. \square

The following is an immediate corollary of Theorems 3.3 and 3.5.

Corollary 3.6 (Undecidability of PHFL Model-Checking Problem). *There is no algorithm that, given a Markov chain M and a closed order-1 formula ϕ of type Prop, decides whether $M \models \phi$.*

We close this section with some remarks.¹

Remark 3.7. Note that the value $\text{val}(A)$ of a probabilistic automaton cannot even be approximately computed [Fij17]: there is no algorithm that outputs “Yes” if $\text{val}(A) = 1$ and “No” if $\text{val}(A) \leq \frac{1}{2}$. Thus, the proof of Theorem 3.5 (in particular, the result $\text{val}(A) = \llbracket \theta_A p_F \rrbracket_{M_A}(q_I)$) also implies that for a qualitative formula of PHFL ψ , $\llbracket \psi \rrbracket$ is not approximately computable in general.

Remark 3.8. It would be interesting to study a converse encoding, i.e., to find an encoding of some fragment of the PHFL model checking problem into the value-1 problem. Such an encoding may help us find a decidable class of the PHFL model checking problem, based on decidable subclasses for the value-1 problem, such as the one studied in [FGKO15].

4. HARDNESS OF THE PHFL MODEL-CHECKING PROBLEM

In the previous section, we have seen that PHFL model checking is undecidable even for the fragment of PHFL without fixpoint alternations. In this section, we give a lower bound of the hardness of the PHFL model-checking problem in the presence of fixpoint alternations. The following theorem states the main result of this section.

Theorem 4.1. *The order-1 PHFL model-checking problem is Π_1^1 -hard and Σ_1^1 -hard.*

¹We would like to thank an anonymous reviewer of our FSCD 2020 submission for pointing them out.

$$\begin{array}{c}
\frac{}{\Gamma, X : A \vdash_{\mu} X : A} \quad \frac{}{\Gamma \vdash_{\mu} Z : N} \quad \frac{\Gamma \vdash_{\mu} s : N}{\Gamma \vdash_{\mu} S s : N} \quad \frac{\Gamma \vdash_{\mu} s : N \quad \Gamma \vdash_{\mu} t : N}{\Gamma \vdash_{\mu} s \leq t : \Omega} \\
\frac{\Gamma \vdash_{\mu} \varphi_1 : \Omega \quad \Gamma \vdash_{\mu} \varphi_2 : \Omega}{\Gamma \vdash_{\mu} \varphi_1 \wedge \varphi_2 : \Omega} \quad \frac{\Gamma \vdash_{\mu} \varphi_1 : \Omega \quad \Gamma \vdash_{\mu} \varphi_2 : \Omega}{\Gamma \vdash_{\mu} \varphi_1 \vee \varphi_2 : \Omega} \quad \frac{\Gamma, X : A \vdash_{\mu} \varphi : T}{\Gamma \vdash_{\mu} \lambda X. \varphi : A \rightarrow T} \\
\frac{\Gamma \vdash_{\mu} \varphi_1 : A \rightarrow T \quad \Gamma \vdash_{\mu} \varphi_2 : A}{\Gamma \vdash_{\mu} \varphi \varphi_2 : T} \quad \frac{\Gamma, X : T \vdash_{\mu} \varphi : T}{\Gamma \vdash_{\mu} \mu X. \varphi : T} \quad \frac{\Gamma, X : T \vdash_{\mu} \varphi : T}{\Gamma \vdash_{\mu} \nu x. \varphi : T}
\end{array}$$

Figure 2: Typing Rules for the Higher-order Fixpoint Arithmetic.

Note that Π_1^1 and Σ_1^1 , defined in terms of the second-order arithmetic, contain very hard problems. For example, those classes contain the problem of deciding whether a given first-order Peano arithmetic formula is true.

We prove this theorem by reducing the validity checking problem of the μ -arithmetic [Lub89] to the PHFL model-checking problem. Even the validity checking problem of a higher-order extension of the μ -arithmetic can be reduced to the PHFL model-checking problem. The key in the proof is a representation of natural numbers as quantitative propositions such that all the operations on natural numbers in the μ -arithmetic are expressible in PHFL.

This section is structured as follows. Section 4.1 reviews the basic notions of the μ -arithmetic. Section 4.2 describes the reduction and proves the theorem above.

4.1. Higher-Order Fixpoint Arithmetic. The μ -arithmetic [Lub89] is a first-order arithmetic with fixpoint operators. This section briefly reviews its higher-order extension, studied by Kobayashi et al. [KTW18].

As in PHFL, we first define the types of μ -arithmetic formulas. The set of *types*, ranged over by A , is given by:

$$A ::= N \mid T \qquad T ::= \Omega \mid A \rightarrow T.$$

The type N is for natural numbers, Ω for (qualitative) propositions, and $A \rightarrow T$ for functions. We do not allow functions to return values of type N . We define the order of types of the μ -arithmetic similarly to the PHFL types, by: $order(N) = order(\Omega) = 0$ and $order(A \rightarrow T) = \max(order(A) + 1, order(T))$.

Assume a countably infinite set Var of variables ranged over by X . The set of formulas, ranged over by φ , is given by the following grammar.

$$s ::= X \mid Z \mid S s \quad \varphi ::= s \mid s_1 \leq s_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \lambda X. \varphi \mid \varphi_1 \varphi_2 \mid \mu X. \varphi \mid \nu X. \varphi.$$

Here, Z and S respectively denote the constant 0 and the successor function on natural numbers.

The typing rules are shown in Fig. 2; they are just standard typing rules for the simply-typed λ -calculus, with several constructors such as $Z : N$, $S : N \rightarrow N$, and $\wedge : \Omega \rightarrow \Omega$. We shall consider only well-typed formulas. We define the *order* of a formula as the largest order of the types of its subformulas.

Definition 4.2 (Semantics of Types). The semantics of a type A is a partially ordered set $\llbracket A \rrbracket_{\mu} = (D_A, \sqsubseteq_A)$ defined inductively on the structure of A as follows.

(1) The semantics of N and Ω :

$$\begin{aligned} D_N &= \mathbb{N} & n \sqsubseteq_N m &\stackrel{\text{def}}{\iff} n = m \\ D_\Omega &= \{0, 1\} & p \sqsubseteq_\Omega q &\stackrel{\text{def}}{\iff} p \leq q \end{aligned}$$

(2) The semantics of $A \rightarrow T$:

$$\begin{aligned} D_{A \rightarrow T} &= \{ f : D_A \rightarrow D_T \mid \forall u, v \in D_A. u \sqsubseteq_A v \implies f(u) \sqsubseteq_T f(v) \} \\ f \sqsubseteq_{A \rightarrow T} g &\stackrel{\text{def}}{\iff} \forall v \in D_A. f(v) \sqsubseteq_T g(v) \end{aligned}$$

The semantics $\llbracket T \rrbracket_\mu$ of a type T forms a complete lattice (while $\llbracket N \rrbracket_\mu$ is not); we write \bigsqcup_T (resp. \bigsqcap_T) for the least upper bound (resp. greatest lower bound) operation, and \perp_T for the least element.

The interpretation $\llbracket \Gamma \rrbracket_\mu$ of a type environment Γ is the set of functions θ such that $\text{dom}(\theta) = \text{dom}(\Gamma)$ and that $\theta(X) \in \llbracket \Gamma(X) \rrbracket_\mu$ for every $X \in \text{dom}(\Gamma)$. It is ordered by the point-wise ordering.

Definition 4.3 (Semantics of Formulas). The semantics of a formula φ with judgment $\Gamma \vdash_\mu \varphi : A$ is a monotone map from $\llbracket \Gamma \rrbracket_\mu$ to $\llbracket A \rrbracket_\mu$, defined as follows.

$$\begin{aligned} \llbracket \Gamma \vdash_\mu X : A \rrbracket_\mu(\theta) &:= \theta(X) \\ \llbracket \Gamma \vdash_\mu Z : N \rrbracket_\mu(\theta) &:= 0 \\ \llbracket \Gamma \vdash_\mu Ss : N \rrbracket_\mu(\theta) &:= \llbracket \Gamma \vdash_\mu s : N \rrbracket_\mu(\theta) + 1 \\ \llbracket \Gamma \vdash_\mu s \leq t : \Omega \rrbracket_\mu(\theta) &:= \begin{cases} 1 & \text{(if } \llbracket \Gamma \vdash_\mu s : N \rrbracket_\mu(\theta) \leq \llbracket \Gamma \vdash_\mu t : N \rrbracket_\mu(\theta) \text{)} \\ 0 & \text{(if } \llbracket \Gamma \vdash_\mu s : N \rrbracket_\mu(\theta) > \llbracket \Gamma \vdash_\mu t : N \rrbracket_\mu(\theta) \text{)} \end{cases} \\ \llbracket \Gamma \vdash_\mu \varphi_1 \wedge \varphi_2 : \Omega \rrbracket_\mu(\theta) &:= \llbracket \Gamma \vdash_\mu \varphi_1 : \Omega \rrbracket_\mu(\theta) \bigsqcap_\Omega \llbracket \Gamma \vdash_\mu \varphi_2 : \Omega \rrbracket_\mu(\theta) \\ \llbracket \Gamma \vdash_\mu \varphi_1 \vee \varphi_2 : \Omega \rrbracket_\mu(\theta) &:= \llbracket \Gamma \vdash_\mu \varphi_1 : \Omega \rrbracket_\mu(\theta) \bigsqcup_\Omega \llbracket \Gamma \vdash_\mu \varphi_2 : \Omega \rrbracket_\mu(\theta) \\ \llbracket \Gamma \vdash_\mu \lambda X. \varphi : A \rightarrow T \rrbracket_\mu(\theta) &:= \lambda v \in \llbracket A \rrbracket_\mu. \llbracket \Gamma, X : A \vdash_\mu \varphi : T \rrbracket_\mu(\theta[X \mapsto v]) \\ \llbracket \Gamma \vdash_\mu \varphi_1 \varphi_2 : T \rrbracket_\mu(\theta) &:= \llbracket \Gamma \vdash_\mu \varphi_1 : A \rightarrow T \rrbracket_\mu(\theta) (\llbracket \Gamma \vdash_\mu \varphi_2 : A \rrbracket_\mu(\theta)) \\ \llbracket \Gamma \vdash_\mu \mu X. \varphi : T \rrbracket_\mu(\theta) &:= LFP(\lambda v \in D_T. \llbracket \Gamma, X : T \vdash_\mu \varphi : T \rrbracket_\mu(\theta[X \mapsto v])) \\ \llbracket \Gamma \vdash_\mu \nu X. \varphi : T \rrbracket_\mu(\theta) &:= GFP(\lambda v \in D_T. \llbracket \Gamma, X : T \vdash_\mu \varphi : T \rrbracket_\mu(\theta[X \mapsto v])) \end{aligned}$$

As in the case of PHFL, we write $\llbracket \varphi \rrbracket_\mu(\theta)$ for $\llbracket \Gamma \vdash_\mu \varphi : A \rrbracket_\mu(\theta)$ and just $\llbracket \varphi \rrbracket_\mu$ for $\llbracket \varphi \rrbracket_\mu(\emptyset)$ when there is no confusion.

Example 4.4. Let $\varphi = \mu F. \lambda X. (X = 100 \vee F(S(SX)))$ where 100 is an abbreviation of the term $\underbrace{S(S(\dots S Z) \dots)}_{100}$. The semantics $\llbracket \varphi \rrbracket_\mu$ is a function $f : \mathbb{N} \rightarrow \{0, 1\}$ where $f(n) = 1$ just if n is an even number no greater than 100.

The *validity checking problem* of the higher-order fixpoint arithmetic is the problem of, given a closed formula φ of type Ω , deciding whether $\llbracket \varphi \rrbracket_\mu = 1$. The following result is probably folklore, which follows from the well-known fact that the *fair termination problem for programs* is Π_1^1 -complete (see, e.g., Harel [Har86]), and the fact that the fair termination of a program can be reduced to the validity of a first-order fixpoint arithmetic formula (see, e.g., [KTW18] for the reduction).

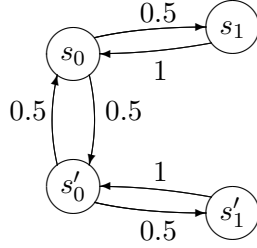


Figure 3: The Markov Chain for Reduction from Higher-order Fixpoint Arithmetic to PHFL.

Theorem 4.5. *The validity checking problem of the first-order fixpoint arithmetic is Π_1^1 -hard and Σ_1^1 -hard.*

Remark 4.6. As for an upper bound, Lubarsky [Lub89] has shown that predicates on natural numbers definable by μ -arithmetic formulas belong to Δ_2^1 . One can prove that the validity problem for the μ -arithmetic is Δ_2^1 as well.

4.2. Hardness of PHFL Model Checking. We give a reduction from the validity checking problem of the higher-order fixpoint arithmetic to the PHFL model-checking problem. The main theorem of this section (Theorem 4.1) is an immediate consequence of this reduction and Theorem 4.5.

Given a formula φ of the higher-order fixpoint arithmetic, we need to effectively construct a pair (ϕ, M) of a formula of PHFL and a Markov chain such that φ is true if and only if $M \models \phi$. The Markov chain M is independent of the formula φ . We first define the Markov chain and then explain the intuition of the translation of formulas.

The Markov chain $M = (S, P, \rho_{AP}, s_{\text{in}})$ is shown in Figure 3. It is defined as follows.

- The set of states is $S = \{s_0, s'_0, s_1, s'_1\}$.
- The transition probability satisfies $P(s_0, s_1) = P(s_0, s'_0) = P(s'_0, s_0) = P(s'_0, s'_1) = \frac{1}{2}$, $P(s_1, s_0) = P(s'_1, s'_0) = 1$ and $P(s_i, s_j) = 0$ for all other pairs of states.
- There are four atomic propositions p_0, p'_0, p_1 , and p'_1 , representing each state (e.g. $\rho_{AP}(p_0) = \{s_0\}$).
- The initial state s_{in} is s_0 .

For notational convenience, we write $v \in \llbracket Prop \rrbracket_M$ as a tuple $(v(s_0), v(s'_0), v(s_1), v(s'_1))$.

As mentioned at the beginning of this section, the key of the reduction is the representation of natural numbers, as well as operations on natural numbers. We encode a propositional formula φ into a quantitative propositional formula ϕ such that $\llbracket \phi \rrbracket_M = (\llbracket \varphi \rrbracket_{\mu, -, -, -})$, and encode a natural number n into a quantitative propositional formula ψ such that $\llbracket \psi \rrbracket_M = (\frac{1}{2^n}, 1 - \frac{1}{2^n}, -, -)$. Here, $-$ denotes a “don’t care” value. We implement primitives on natural numbers Z , S and \leq , as follows.

The constant Z can be represented by p_0 . Indeed, $\llbracket p_0 \rrbracket_M = (1, 0, 0, 0) = (1/2^0, 1 - (1/2^0), 0, 0)$ as expected.

Assuming that ϕ represents n (i.e. $\llbracket \phi \rrbracket_M = (1/2^n, 1 - (1/2^n), -, -)$), the successor $n + 1$ can be represented by

$$\phi' := \bigcirc((\bigcirc\phi \wedge (p_1 \vee p'_1)) \vee p_0).$$

Indeed, we have:

$$\begin{aligned}
\llbracket \circ\phi \rrbracket_M &= (-, -, \frac{1}{2^n}, 1 - \frac{1}{2^n}) \\
\llbracket \circ\phi \wedge (p_1 \vee p'_1) \rrbracket_M &= (0, 0, \frac{1}{2^n}, 1 - \frac{1}{2^n}) \\
\llbracket (\circ\phi \wedge (p_1 \vee p'_1)) \vee p_0 \rrbracket_M &= (1, 0, \frac{1}{2^n}, 1 - \frac{1}{2^n}) \\
\llbracket \circ((\circ\phi \wedge (p_1 \vee p'_1)) \vee p_0) \rrbracket_M &= (\frac{1}{2} \times \frac{1}{2^n}, \frac{1}{2} + \frac{1}{2} \times (1 - \frac{1}{2^n}), -, -) \\
&= (\frac{1}{2^{n+1}}, 1 - \frac{1}{2^{n+1}}, -, -).
\end{aligned}$$

It remains to encode \leq . We use the fact that, for any natural numbers n and m ,

$$n \leq m \iff \frac{1}{2^n} \geq \frac{1}{2^m} \iff \frac{1}{2^n} + (1 - \frac{1}{2^m}) \geq 1.$$

The s'_0 -component of the representation of a natural number plays an important role below. Assume that ϕ and χ represent n and m respectively. Then we have

$$\llbracket \circ\phi \wedge p_1 \rrbracket_M = (0, 0, \frac{1}{2^n}, 0) \quad \llbracket \chi \wedge p'_0 \rrbracket_M = (0, 1 - \frac{1}{2^m}, 0, 0)$$

and thus

$$\llbracket (\circ\phi \wedge p_1) \vee (\chi \wedge p'_0) \rrbracket_M = (0, 1 - \frac{1}{2^m}, \frac{1}{2^n}, 0).$$

Therefore

$$\llbracket \circ((\circ\phi \wedge p_1) \vee (\chi \wedge p'_0)) \rrbracket_M = (\frac{1}{2} \times (\frac{1}{2^n} + (1 - \frac{1}{2^m})), -, -, -).$$

Thus, $n \leq m$ if and only if the s_0 -component of the above formula is $\geq \frac{1}{2}$. In other words, $n \leq m$ just if:

$$\llbracket \llbracket \circ((\circ\phi \wedge p_1) \vee (\chi \wedge p'_0)) \rrbracket_{\frac{1}{2}} \rrbracket_M = (1, -, -, -).$$

Let us formalize the above argument. We first give the translation of types:

$$tr(N) = Prop \quad tr(\Omega) = Prop \quad tr(A \rightarrow T) = tr(A) \rightarrow tr(T).$$

The translation can be naturally extended to type environments. Following the above discussion, the translation of formulas of type N is given by

$$tr(Z) = p_0 \quad \text{and} \quad tr(S s) = \circ((\circ tr(s) \wedge (p_1 \vee p'_1)) \vee p_0).$$

The comparison operator can be translated as follows:

$$tr(s \leq t) = \llbracket \circ((\circ tr(s) \wedge p_1) \vee (tr(t) \wedge p'_0)) \rrbracket_{\geq \frac{1}{2}}.$$

The translation of other connectives is straightforward:

$$\begin{aligned}
tr(\varphi_1 \wedge \varphi_2) &= tr(\varphi_1) \wedge tr(\varphi_2) & tr(\varphi_1 \vee \varphi_2) &= tr(\varphi_1) \vee tr(\varphi_2) & tr(\lambda X. \varphi) &= \lambda X. tr(\varphi) \\
tr(X) &= X & tr(\varphi_1 \varphi_2) &= tr(\varphi_1) tr(\varphi_2) & tr(\mu X. \varphi) &= \mu X. tr(\varphi) & tr(\nu X. \varphi) &= \nu X. tr(\varphi).
\end{aligned}$$

The following lemma states that the translation preserves types.

Lemma 4.7. *If $\Gamma \vdash_{\mu} \varphi : A$, then $tr(\Gamma) \vdash tr(\varphi) : tr(A)$.*

Proof. This follows by straightforward induction on the derivation of $\Gamma \vdash_{\mu} \varphi : A$. □

We prove the correctness of the translation. For each type A of the higher-order fixpoint arithmetic, we define a relation $(\sim_A) \subseteq \llbracket A \rrbracket_\mu \times \llbracket tr(A) \rrbracket_M$ by induction on A as follows:

$$\begin{aligned} n \sim_N (r_0, r'_0, r_1, r'_1) &\stackrel{\text{def}}{\iff} r_0 = \frac{1}{2^n} \text{ and } r'_0 = 1 - \frac{1}{2^n} \\ b \sim_\Omega (r_0, r'_0, r_1, r'_1) &\stackrel{\text{def}}{\iff} b = r_0 \\ f \sim_{A \rightarrow T} g &\stackrel{\text{def}}{\iff} \forall x \in \llbracket A \rrbracket_\mu. \forall y \in \llbracket tr(A) \rrbracket_M. x \sim_A y \implies f x \sim_T g y. \end{aligned}$$

This relation can be naturally extended to the interpretations of type environments: given a type environment Γ of the μ -arithmetic, the relation $(\sim_\Gamma) \subseteq \llbracket \Gamma \rrbracket_\mu \times \llbracket tr(\Gamma) \rrbracket_M$ is defined by

$$\theta \sim_\Gamma \rho \stackrel{\text{def}}{\iff} \forall X \in \text{dom}(\Gamma). \theta(X) \sim_{\Gamma(X)} \rho(X).$$

The following theorem states the correspondence between the source and the target of the translation. A proof is provided in Appendix A.

Theorem 4.8. *Let $\Gamma \vdash_\mu \varphi : A$ be a formula of the higher-order fixpoint arithmetic. Assume $\theta \in \llbracket \Gamma \rrbracket_\mu$ and $\rho \in \llbracket tr(\Gamma) \rrbracket_M$. If $\theta \sim_\Gamma \rho$, then $\llbracket \Gamma \vdash_\mu \varphi : A \rrbracket_\mu(\theta) \sim_A \llbracket tr(\Gamma) \vdash tr(\varphi) : tr(A) \rrbracket_M(\rho)$.*

Corollary 4.9. *The validity problem of the order- k fixpoint arithmetic is reducible to the order- k PHFL model-checking problem.*

Proof. Assume $\emptyset \vdash_\mu \varphi : \Omega$. By Theorem 4.8, $\llbracket \varphi \rrbracket_\mu \sim_\Omega \llbracket tr(\varphi) \rrbracket_M$. Therefore, $\llbracket \varphi \rrbracket_\mu = 1$ if and only if $\llbracket tr(\varphi) \rrbracket_M(s_0) = 1$, i.e. $M \models tr(\varphi)$. The mapping $\varphi \mapsto (tr(\varphi), M)$ is obviously effective, and preserves the order. \square

Theorem 4.1 is an immediate consequence of Theorem 4.5 and Corollary 4.9.

5. DECIDABLE SUBCLASS OF ORDER-1 PHFL MODEL CHECKING

As we have seen in Section 3, PHFL model checking is undecidable, even for order 1. In this section, we identify a decidable subclass of the order-1 PHFL model-checking problems (i.e., a set of pairs (ϕ, M) such that whether $M \models \phi$ is decidable). We identify the subclass by using a type system: we define a type system \mathcal{T}_M for PHFL formulas, parameterized by M , so that if ϕ is a proposition well-typed in \mathcal{T}_M , then $M \models \phi$ is decidable.

This section is structured as follows. In Section 5.1, we introduce the type system \mathcal{T}_M , and prove that the semantics of any order-1 well-typed formula is an affine function. Section 5.2 introduces a matrix representation of affine functions and shows the decidability of $M \models \phi$ by appealing to the decidability of the first-order theory of reals [Tar51]. Section 5.3 shows that the restricted fragment is reasonably expressive, by giving an encoding of the termination problem for recursive Markov chains into the restricted fragment of PHFL model checking.

5.1. Type-based Restriction of Order-1 PHFL. We first explain the idea of the restriction imposed by our type system. By definition, the semantics of a (closed) order-1 PHFL formula ϕ of type $Prop \rightarrow Prop$ with respect to the Markov chain M is a map f_ϕ from the set of functions $S \rightarrow [0, 1]$ to the same set, where S is the set of states of M . Thus, if $S = \{s_1, s_2, \dots, s_n\}$ is fixed, f_ϕ can be regarded as a function from $[0, 1]^n$ to $[0, 1]^n$. Now, if the function f_ϕ were affine, i.e., if there are functions f_1, f_2, \dots, f_n such that $f_\phi(r_1, r_2, \dots, r_n) = (f_1(r_1, r_2, \dots, r_n), \dots, f_n(r_1, r_2, \dots, r_n))$, where

$f_i(r_1, r_2, \dots, r_n) = c_{i,0} + c_{i,1}r_1 + \dots + c_{i,n}r_n$ for some real numbers $c_{i,j}$, then the function f_ϕ would be representable by a finite number of reals $c_{i,j}$. The semantics of a fixpoint formula would then be given as a solution of a fixpoint equation on the coefficients, which is solvable by appealing to the decidability of first-order theories of reals [Tar51].

Based on the observation above, we introduce a type system to restrict the formulas so that the semantics of every well-typed order-1 formula is affine. The conjunction $\phi_1 \wedge \phi_2$ is one of the problematic logical connectives that may make the semantics of an order-1 formula non-affine: recall that the min operator was used to define the semantics of conjunction. We require that for every subformula of the form $\phi_1 \wedge \phi_2$ and for each state $s \in S$, one of the values $\llbracket \phi_1 \rrbracket(s)$ and $\llbracket \phi_2 \rrbracket(s)$ is the constant 0 or 1. We can then remove the min operator, since we have $\min(0, x) = 0$ and $\min(1, x) = x$ for every $x \in [0, 1]$.

We parameterize the type system by the Markov chain M , since it often depends on M whether the semantics of an order-1 formula is affine. For example, the semantics of $(p \wedge \phi_1) \vee (q \wedge \phi_2)$ is affine if the semantics of ϕ_1 and ϕ_2 are affine *and if* p and q cannot be simultaneously true (i.e., if $\rho_{AP,M}(p) \cap \rho_{AP,M}(q) = \emptyset$). Without the parameterization, the resulting type system would be too conservative.

The discussion above motivates us to refine the type $Prop$ of propositions to $Prop^{T,U}$ where $T, U \subseteq S$ and $T \cap U = \emptyset$. Intuitively, the type $Prop^{T,U}$ describes propositions $\phi \in Prop$ such that $\llbracket \phi \rrbracket(s) = 0$ for all $s \in T$ and $\llbracket \phi \rrbracket(s) = 1$ for all $s \in U$; there is no guarantee on the value of $\llbracket \phi \rrbracket(s)$ for $s \in S \setminus (T \cup U)$. The syntax of *refined types* is given by:

$$\kappa ::= Prop^{T,U} \mid Prop^{T,U} \rightarrow \kappa$$

where T and U range over the set of subsets of S satisfying $T \cap U = \emptyset$. Note that each type κ can be expressed as $Prop^{T_1, U_1} \rightarrow Prop^{T_2, U_2} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U}$ where $k \geq 0$.

We define the translation from the set of refined types to the set of types in PHFL by

$$tr(Prop^{T,U}) = Prop \qquad tr(\kappa_1 \rightarrow \kappa_2) = tr(\kappa_1) \rightarrow tr(\kappa_2)$$

and the translation of type environment \mathcal{K} by $(tr(\mathcal{K}))(x) = tr(\mathcal{K}(x))$. The semantics of refined types is defined as follows. As explained above, the values of function types are restricted to affine functions.

Definition 5.1. For each refined type κ , we define the subset $\llbracket \kappa \rrbracket \subseteq \llbracket tr(\kappa) \rrbracket$ as follows.

$$\begin{aligned} \llbracket Prop^{T,U} \rrbracket &= \{v \in D_{Prop} \mid \forall s \in T. v(s) = 0, \forall s \in U. v(s) = 1\} \\ \llbracket Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U} \rrbracket &= \\ &\{f \in \llbracket Prop^k \rightarrow Prop \rrbracket \mid \\ &\quad f \text{ is affine on } \llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_k, U_k} \rrbracket, \text{ and} \\ &\quad \forall v_1 \in \llbracket Prop^{T_1, U_1} \rrbracket, \dots, v_k \in \llbracket Prop^{T_k, U_k} \rrbracket. f v_1 \dots v_k \in \llbracket Prop^{T, U} \rrbracket\} \end{aligned}$$

In the definition above, by “ f is affine on $D_1 \times \dots \times D_k$ ”, we mean that, for each state $s_\ell \in S$, there exist some coefficients $c_0^\ell, c_{1,1}^\ell, \dots, c_{k,n}^\ell$ such that, for every $(v_1, \dots, v_k) \in D_1 \times \dots \times D_k$,

$$f v_1 \dots v_k s_\ell = c_0^\ell + \sum_{i \in \{1, \dots, k\}, j \in \{1, \dots, n\}} c_{i,j}^\ell v_i(s_j).$$

Remark 5.2. Note that $\llbracket \kappa \rrbracket$ is not closed under various operations. For example, the greatest lower bound of affine functions $\lambda(x, y).x$ and $\lambda(x, y).y \in [0, 1]^2 \rightarrow [0, 1]$ is $\lambda(x, y). \min(x, y)$, which is not affine. This means that the conjunction does not preserve affinity, as mentioned above. A similar observation applies to fixpoints: for a monotone function h on $\llbracket tr(\kappa) \rrbracket$, even if $h x \in \llbracket \kappa \rrbracket$ for every $x \in \llbracket \kappa \rrbracket$, it is not necessarily the case that $LFP(h) \in \llbracket \kappa \rrbracket$. For

example, let $S = \{s\}$, $\kappa = Prop^{\emptyset, \emptyset} \rightarrow Prop^{\emptyset, \{s\}}$, and $h(f) = \lambda v. \lambda s. \max(f(v)(s), v(s)^2)$. For any $f \in \llbracket \kappa \rrbracket$ and $v \in Prop^{\emptyset, \emptyset}$, $h(f)(v)(s) = \max(f(v)(s), v(s)^2) = \max(1, v(s)^2) = 1$, hence $h(f) \in \llbracket \kappa \rrbracket$. However, $LFP(h) = \lambda v. \lambda s. v(s)^2 \notin \llbracket \kappa \rrbracket$.

We restrict PHFL formulas by a type system parameterized by a Markov chain M . We consider a type judgment of the form: $\mathcal{K}; \Delta \vdash_M \phi : \kappa$. Here, \mathcal{K} is a type environment of the form $X_1 : \kappa_1, \dots, X_k : \kappa_k$; it is for fixpoint variables, i.e., those bound by μ or ν . The other type environment Δ is of the form $Y_1 : Prop^{T_1, U_1}, \dots, Y_m : Prop^{T_\ell, U_\ell}$; it is for variables bound by λ . We require that the domains of \mathcal{K} and Δ are disjoint. The intended meaning of the judgment $\mathcal{K}; \Delta \vdash_M \phi : \kappa$, where $\Delta = Y_1 : Prop^{T_1, U_1}, \dots, Y_\ell : Prop^{T_\ell, U_\ell}$ and $\kappa = Prop^{T_{\ell+1}, U_{\ell+1}} \rightarrow \dots \rightarrow Prop^{T_{\ell+m}, U_{\ell+m}} \rightarrow Prop^{T, U}$ is as follows. Assume: (i) each fixpoint variable X is bound to an affine function as described by $\mathcal{K}(X)$, (ii) each Y_i ($1 \leq i \leq \ell + m$) is bound to a value $(x_{i,1}, \dots, x_{i,n})$ described by $Prop^{T_i, U_i}$. Then the value of $\phi Y_{\ell+1} \cdots Y_{\ell+m}$ is an affine function on $x_{i,j}$. Note that the value of $\phi Y_{\ell+1} \cdots Y_{\ell+m}$ need not be affine on the values of fixpoint variables. Below Δ is treated as a *sequence* of type bindings, while \mathcal{K} is treated as a *set*.

The typing rules are given in Figure 4. We explain key rules below. The rule T-WEAKTU is for weakening the information represented by T and U ; this rule is required, for example, for adjusting the types between a function and its argument. The rule T-WEAK is a usual weakening rule for adding type bindings to Δ . The rule T-AP is for atomic propositions; recall that $\rho_{AP}(p)$ denotes the set of states where p holds with probability 1. The rule T-MU is for least fixpoint formulas. The second premise means that κ is of the form $Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U}$, where $U = \emptyset$.² Without this restriction, the value of $\phi \phi_1 \cdots \phi_k$ at a state in U may be wrongly estimated to be 1. For example, consider the case where $\phi = X$ and the simple type of X is $Prop$. Then, the value of $\mu X.X$ should be the map f such that $f(s) = 0$ for every state. Without the restriction, however, we could wrongly derive $\mu X.X : Prop^{\emptyset, S_M}$. Note also that Δ is empty in T-MU; this is just for technical convenience, and is not a fundamental restriction. Indeed, if $\mu X.\phi$ contains a free variable Y of type $Prop^{T, U}$, then we can replace it with $(\mu X'. \lambda Y. [X' Y / X] \phi) Y$, without changing the semantics. Analogous conditions are imposed in the rule T-NU for greatest fixpoint formulas. In the rule T-CONJ for conjunctions, the first two premises imply that the value of ϕ_i at a state in T_i is 0; therefore, the value of $\phi_1 \wedge \phi_2$ at a state in $T_1 \cup T_2$ is 0, which explains $T_1 \cup T_2$ in the conclusion. Similarly for $U_1 \cap U_2$. The third premise (on the second line) ensures that the value of $\phi_1 \wedge \phi_2$ is an affine function on the value of the variables in Δ . That is guaranteed if $\Delta = \emptyset$. Otherwise, we require $T_1 \cup U_1 \cup T_2 \cup U_2 = S_M$; recall the earlier discussion on a sufficient condition for the semantics of an order-1 formula to be affine. The rule T-DISJ for disjunctions is analogous. In the rules T-J, T-MIN, and T-MAX, we require that the type environment Δ for λ -bound variables be empty, since the operators $[\cdot]_J, \square$, and \diamond break the affinity. The sets T' and U' in the conclusions of those rules are conservatively approximated. In T-J, recall that we have excluded out trivial

²The condition $U = \emptyset$ is sometimes too restrictive. For example, consider the formula $\mu X.\text{true}$. We can only assign $Prop^{\emptyset, \emptyset}$, although $Prop^{\emptyset, S_M}$ can be assigned to the equivalent formula true . To remedy this problem, it suffices to add the rule for unfolding:

$$\frac{\mathcal{K}; \Delta \vdash_M [\mu X.\phi / X] \phi : \kappa}{\mathcal{K}; \Delta \vdash_M \mu X.\phi : \kappa}$$

For the sake of simplicity, we do not consider this rule.

$$\begin{array}{c}
\frac{\mathcal{K}; \Delta \vdash_M \phi : Prop^{T,U}}{T' \subseteq T \quad U' \subseteq U} \\
\frac{\mathcal{K}; \Delta \vdash_M \phi : Prop^{T',U'}}{\quad} \quad (T\text{-WEAKTU})
\end{array}
\qquad
\frac{\mathcal{K}; \Delta_1, X : Prop^{T,U}, Y : Prop^{T',U'}, \Delta_2 \vdash_M \phi : \kappa}{\mathcal{K}; \Delta_1, Y : Prop^{T',U'}, X : Prop^{T,U}, \Delta_2 \vdash_M \phi : \kappa} \quad (T\text{-EX})$$

$$\frac{\mathcal{K}; \Delta \vdash_M \phi : \kappa}{\mathcal{K}; X : Prop^{T,U}, \Delta \vdash_M \phi : \kappa} \quad (T\text{-WEAK})
\qquad
\frac{\mathcal{K}; \Delta \vdash_M \phi_1 : Prop^{T_1, U_1} \quad \mathcal{K}; \Delta \vdash_M \phi_2 : Prop^{T_2, U_2}}{\Delta \neq \emptyset \Rightarrow T_1 \cup U_1 \cup T_2 \cup U_2 = S_M} \quad (T\text{-CONJ})$$

$$\frac{}{\mathcal{K}; \emptyset \vdash_M p : Prop^{\rho_{AP}(p), \rho_{AP}(p)}} \quad (T\text{-AP})
\qquad
\frac{\mathcal{K}; \Delta \vdash_M \phi_1 : Prop^{T_1, U_1} \quad \mathcal{K}; \Delta \vdash_M \phi_2 : Prop^{T_2, U_2}}{\Delta \neq \emptyset \Rightarrow T_1 \cup U_1 \cup T_2 \cup U_2 = S_M} \quad (T\text{-CONJ})$$

$$\frac{}{\mathcal{K}, X : \kappa; \emptyset \vdash_M X : \kappa} \quad (T\text{-FVAR})
\qquad
\frac{\mathcal{K}; \Delta \vdash_M \phi_1 : Prop^{T_1, U_1} \quad \mathcal{K}; \Delta \vdash_M \phi_2 : Prop^{T_2, U_2}}{\Delta \neq \emptyset \Rightarrow T_1 \cap T_2, (U_1 \cup U_2)} \quad (T\text{-DISJ})$$

$$\frac{}{\mathcal{K}; X : Prop^{T,U} \vdash_M X : Prop^{T,U}} \quad (T\text{-VAR})
\qquad
\frac{\mathcal{K}; \emptyset \vdash_M \phi : Prop^{T,U}}{\mathcal{K}; \emptyset \vdash_M [\phi]_J : Prop^{T,U}} \quad (T\text{-J})$$

$$\frac{\mathcal{K}, X : \kappa; \emptyset \vdash_M \phi : \kappa}{\kappa \text{ is of the form } \dots \rightarrow Prop^{T, \emptyset}} \quad (T\text{-MU})
\qquad
\frac{\mathcal{K}; \emptyset \vdash_M \phi : Prop^{T,U}}{T' = \{s \in S_M \mid \exists s' \in T. P_M(s, s') > 0\}} \quad (T\text{-MIN})$$

$$\frac{\mathcal{K}, X : \kappa; \emptyset \vdash_M \phi : \kappa}{\kappa \text{ is of the form } \dots \rightarrow Prop^{\emptyset, U}} \quad (T\text{-NU})
\qquad
\frac{\mathcal{K}; \emptyset \vdash_M \phi : Prop^{T,U}}{T' = \{s \in S_M \mid \forall s' \in S_M. P_M(s, s') > 0 \Rightarrow s' \in T\}} \quad (T\text{-MAX})$$

$$\frac{\mathcal{K}; \Delta, X : Prop^{T,U} \vdash_M \phi : \kappa}{\mathcal{K}; \Delta \vdash_M \lambda X. \phi : Prop^{T,U} \rightarrow \kappa} \quad (T\text{-ABS})
\qquad
\frac{\mathcal{K}; \emptyset \vdash_M \phi : Prop^{T,U}}{U' = \{s \in S_M \mid \exists s' \in U. P_M(s, s') > 0\}} \quad (T\text{-MAX})$$

$$\frac{\mathcal{K}; \Delta \vdash_M \phi_1 : Prop^{T,U} \rightarrow \kappa \quad \mathcal{K}; \Delta \vdash_M \phi_2 : Prop^{T,U}}{\mathcal{K}; \Delta \vdash_M \phi_1 \phi_2 : \kappa} \quad (T\text{-APP})
\qquad
\frac{\mathcal{K}; \Delta \vdash_M \phi : Prop^{T,U}}{T' = \{s \in S_M \mid \forall s' \in S_M. P_M(s, s') > 0 \Rightarrow s' \in T\}} \quad (T\text{-MAX})$$

$$\frac{\mathcal{K}; \Delta \vdash_M \phi_1 \phi_2 : \kappa}{\quad} \quad (T\text{-APP})
\qquad
\frac{\mathcal{K}; \Delta \vdash_M \phi : Prop^{T,U}}{U' = \{s \in S_M \mid \forall s' \in S_M. P_M(s, s') > 0 \Rightarrow s' \in U\}} \quad (T\text{-MAX})$$

$$\frac{}{\mathcal{K}; \Delta \vdash_M \phi_1 \phi_2 : \kappa} \quad (T\text{-APP})
\qquad
\frac{}{\mathcal{K}; \Delta \vdash_M \bigcirc \phi : Prop^{T', U'}} \quad (T\text{-AVG})$$

Figure 4: Typing Rules for the Decidable Fragment

bounds such as > 1 and ≥ 0 ; thus, the value of $[\phi]_J$ is 0 (1, resp.) if the value of ϕ is 0 (1, resp.). In the rule T-AVG, we need not require Δ to be empty, as the average of affine functions is again affine.

Example 5.3. Let M be an element of \mathcal{M} in the proof of Theorem 2.9, i.e., a Markov chain $(S, P, \rho_{AP}, s_{\text{in}})$ that satisfies the following conditions.

- $S = \{s_0, s_1, \dots, s_n\}$ for a positive integer n ,
- $P(s_i, s_{i+1}) = 1$ ($0 \leq i \leq n-1$), $P(s_n, s_n) = 1$ and $P(s_i, s_j) = 0$ otherwise.
- There are three atomic propositions a, b, c with $\rho_{AP}(a) \cup \rho_{AP}(b) = \{s_0, s_1, \dots, s_{n-1}\}$, $\rho_{AP}(a) \cap \rho_{AP}(b) = \emptyset$ and $\rho_{AP}(c) = \{s_n\}$.
- The initial state is $s_{\text{in}} = s_0$.

Let ϕ_1 be the formula:

$$(\mu F. \lambda X. a \wedge \bigcirc(X \vee F(b \wedge \bigcirc X)))(b \wedge \diamond c),$$

which is a variation of the formula ϕ considered in the proof, obtained by replacing two occurrences of \diamond with \bigcirc . Since M has only deterministic transitions, ϕ_1 has the same value as ϕ . Let $\mathcal{K} = F : Prop^{\rho_{AP}(a), \emptyset} \rightarrow Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}$ and $\Delta = X : Prop^{\rho_{AP}(a), \emptyset}$. Then, we have:

$$\frac{\frac{\frac{\mathcal{K}; \Delta \vdash_M a : Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \rho_{AP}(a)}}{\mathcal{K}; \Delta \vdash_M a \wedge \bigcirc(X \vee F(b \wedge \bigcirc X)) : Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}}{\mathcal{K}; \Delta \vdash_M X : Prop^{\rho_{AP}(a), \emptyset}} \quad \frac{\mathcal{K}; \Delta \vdash_M F(b \wedge \bigcirc X) : Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}}{\mathcal{K}; \Delta \vdash_M X \vee F(b \wedge \bigcirc X) : Prop^{\emptyset, \emptyset}} \quad \dots}{\mathcal{K}; \Delta \vdash_M \bigcirc(X \vee F(b \wedge \bigcirc X)) : Prop^{\emptyset, \emptyset}}{\mathcal{K}; \Delta \vdash_M a \wedge \bigcirc(X \vee F(b \wedge \bigcirc X)) : Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}}$$

Here, $\mathcal{K}; \Delta \vdash_M F(b \wedge \bigcirc X) : Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}$ is derived as follows.

$$\frac{\frac{\frac{\mathcal{K}; \Delta \vdash_M b : Prop^{\rho_{AP}(a) \cup \rho_{AP}(c), \rho_{AP}(b)}}{\mathcal{K}; \Delta \vdash_M b \wedge \bigcirc X : Prop^{\rho_{AP}(a) \cup \rho_{AP}(c), \emptyset}}{\mathcal{K}; \Delta \vdash_M F : Prop^{\rho_{AP}(a), \emptyset} \rightarrow Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}} \quad \frac{\mathcal{K}; \Delta \vdash_M X : Prop^{\rho_{AP}(a), \emptyset}}{\mathcal{K}; \Delta \vdash_M \bigcirc X : Prop^{\emptyset, \emptyset}}}{\mathcal{K}; \Delta \vdash_M b \wedge \bigcirc X : Prop^{\rho_{AP}(a), \emptyset}}}{\mathcal{K}; \Delta \vdash_M F(b \wedge \bigcirc X) : Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}}$$

We can thus obtain

$$\emptyset; \emptyset \vdash_M (\mu F. \lambda X. a \wedge \bigcirc(X \vee F(b \wedge \bigcirc X)))(b \wedge \diamond c) : Prop^{\rho_{AP}(b) \cup \rho_{AP}(c), \emptyset}.$$

Note that, by the same argument as the proof of Theorem 2.9, there exists no μ^p -calculus formula equivalent to ϕ_1 . \square

The following lemma states that a formula that is well-typed in \mathcal{T}_M is also well-typed in the original PHFL type system.

Lemma 5.4. *Let ϕ be a PHFL formula such that $\mathcal{K}; \Delta \vdash_M \phi : \kappa$. Then we have $tr(\mathcal{K}, \Delta) \vdash \phi : tr(\sigma)$.*

Proof. This follows by a straightforward induction on the derivation of $\mathcal{K}; \Delta \vdash_M \phi : \kappa$. \square

The following lemma states that the refined type system does not impose any restriction on the order-0 fragment of PHFL. Thus, together with the observation in Example 5.3, the lemma implies that our decidable fragment is strictly more expressive than the μ^p -calculus.

Lemma 5.5. *If $\Gamma \vdash \phi : \tau$, and ϕ is an order-0 formula, then $\mathcal{K}; \emptyset \vdash_M \phi : Prop^{\emptyset, \emptyset}$ where \mathcal{K} is the type environment such that $dom(\mathcal{K}) = dom(\Gamma)$ and $\mathcal{K}(X) = Prop^{\emptyset, \emptyset}$ for every $X \in dom(\mathcal{K})$.*

Proof. This follows by a straightforward induction on the derivation of $\Gamma \vdash \phi : \tau$. Note that since Δ is always empty, the condition $T_1 \cup U_1 \cup T_2 \cup U_2$ in T-CONJ and T-DISJ is irrelevant. \square

In the rest of this subsection, we prove the following properties.

- (1) The type system is sound in the sense that the semantics of any formula ϕ of type κ indeed belongs to $\llbracket \kappa \rrbracket$; see Theorem 5.8 for the precise statement.

- (2) The calculation of the semantics of a well-typed formula (especially, the least/greatest fixpoint computation) can be performed up to the equivalence relation \sim_κ , where $f \sim_{Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow Prop^{T, U}} g$ just if f and g are equivalent on the intended domain, i.e., if $f v_1 \cdots v_m = g v_1 \cdots v_m$ for any $v_1 \in \llbracket Prop^{T_1, U_1} \rrbracket, \dots, v_m \in \llbracket Prop^{T_m, U_m} \rrbracket$; see Lemmas 5.10 and 5.11.

The reason why the type system ensures affinity has been intuitively explained already, except for the fixpoints. Here we show (in Lemma 5.7) that the fixpoint of a typable fixpoint operator is indeed affine. The key observation is that $\llbracket \kappa \rrbracket \subseteq \llbracket tr(\kappa) \rrbracket$ is closed under the limit of chains, as stated in the following lemma.

Lemma 5.6. *Let κ be a refined type and γ be an ordinal number. For every increasing chain $(f_\alpha)_{\alpha < \gamma}$ of elements in $\llbracket \kappa \rrbracket$, the limit $\bigsqcup_{\alpha < \gamma} f_\alpha$ in $\llbracket tr(\kappa) \rrbracket$ belongs to $\llbracket \kappa \rrbracket$. Similarly, for every decreasing chain $(f_\alpha)_{\alpha < \gamma}$, the limit $\bigsqcap_{\alpha < \gamma} f_\alpha$ is in $\llbracket \kappa \rrbracket$.*

Proof. We prove the former. Assume $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U}$.

We first give an alternative characterization of affinity. For each $i \leq k$, given $v_i \in \llbracket Prop^{T_i, U_i} \rrbracket$ and $r \in [0, 1]$, we define $r \cdot v_i$ by $(r \cdot v_i)(s) = r(v_i(s))$. Note that $r \cdot v_i$ may not be a member of $\llbracket Prop^{T_i, U_i} \rrbracket$, but $r \cdot v_i + (1 - r) \cdot v'_i \in \llbracket Prop^{T_i, U_i} \rrbracket$ for every $v_i, v'_i \in \llbracket Prop^{T_i, U_i} \rrbracket$ and $r \in [0, 1]$ (here the sum is the point-wise sum on reals). Then $f \in \llbracket Prop^k \rightarrow Prop \rrbracket$ is affine on $\llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_k, U_k} \rrbracket$ if and only if, for every $(v_1, \dots, v_k), (v'_1, \dots, v'_k) \in \llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_k, U_k} \rrbracket$ and $r \in [0, 1]$,

$$f(r \cdot v_1 + (1 - r) \cdot v'_1) \dots (r \cdot v_k + (1 - r) \cdot v'_k) = r(f v_1 \dots v_k) + (1 - r)(f v'_1 \dots v'_k).$$

Let $(f_\alpha)_{\alpha < \gamma}$ be an increasing chain and $f = \bigsqcup_{\alpha < \gamma} f_\alpha$. Then f can be characterized in terms of the limits in real numbers as $f y_1 \dots y_k = \lim_{\alpha < \gamma} (f_\alpha y_1 \dots y_k)$ for every $y_1, \dots, y_k \in \llbracket Prop \rrbracket$. Since $\lim_{\alpha < \gamma}$ commutes with linear operations, for every $(v_1, \dots, v_k), (v'_1, \dots, v'_k) \in \llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_k, U_k} \rrbracket$ and $r \in [0, 1]$, we have:

$$\begin{aligned} & f(r \cdot v_1 + (1 - r) \cdot v'_1) \dots (r \cdot v_k + (1 - r) \cdot v'_k) \\ &= \lim_{\alpha < \gamma} (f_\alpha (r \cdot v_1 + (1 - r) \cdot v'_1) \dots (r \cdot v_k + (1 - r) \cdot v'_k)) \\ &= \lim_{\alpha < \gamma} (r(f_\alpha v_1 \dots v_k) + (1 - r)(f_\alpha v'_1 \dots v'_k)) \\ &= r \left(\lim_{\alpha < \gamma} (f_\alpha v_1 \dots v_k) \right) + (1 - r) \left(\lim_{\alpha < \gamma} (f_\alpha v'_1 \dots v'_k) \right) \\ &= r(f v_1 \dots v_k) + (1 - r)(f v'_1 \dots v'_k). \end{aligned}$$

The latter is the dual of the former, and can be proved in the same manner, by just replacing \bigsqcup with \bigsqcap . \square

Lemma 5.7. *Let $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U}$ be a refined type and h be a monotone function on $tr(\kappa)$ such that $x \in \llbracket \kappa \rrbracket$ implies $h x \in \llbracket \kappa \rrbracket$.*

- If $U = \emptyset$, then $LFP(h) \in \llbracket \kappa \rrbracket$.
- If $T = \emptyset$, then $GFP(h) \in \llbracket \kappa \rrbracket$.

Here LFP and GFP are taken in $\llbracket tr(\kappa) \rrbracket$.

Proof. We prove the former; the latter can be proved similarly. We define $h^\alpha(\perp)$ for ordinals α as follows:

$$h^\alpha(\perp) = \begin{cases} \perp_{tr(\kappa)} & \text{if } \alpha = 0 \\ h(h^\beta(\perp)) & \text{if } \alpha = \beta + 1 \\ \bigsqcup_{\beta < \alpha} h^\beta(\perp) & \text{if } \alpha \text{ is a limit ordinal.} \end{cases}$$

Then $LFP(h) = h^\alpha(\perp)$ for some sufficiently large α .

It suffices to show that $h^\alpha(\perp) \in \llbracket \kappa \rrbracket$ for every α . We prove this claim by transfinite induction. For $\alpha = 0$, $\perp_{tr(\kappa)} \in \llbracket \kappa \rrbracket$ since $U = \emptyset$. For $\alpha = \beta + 1$, we have $h(h^\beta(\perp)) \in \llbracket \kappa \rrbracket$ since $h^\beta(\perp) \in \llbracket \kappa \rrbracket$. If α is a limit cardinal, we appeal to Lemma 5.6; note that $(f^\beta(\perp))_{\beta < \alpha}$ is an increasing chain. \square

Note that in the above lemma, $LFP(h) \in \llbracket \kappa \rrbracket$ would not hold if we drop the condition $U = \emptyset$; recall Remark 5.2. That justifies the corresponding conditions in rule T-MU and T-NU.

The following theorem is soundness of the type system. Let $\llbracket \mathcal{K} \rrbracket$ be the subset of $\llbracket tr(\mathcal{K}) \rrbracket$ consisting of interpretations ρ such that $\rho(X) \in \llbracket \mathcal{K}(X) \rrbracket$ for every $X \in dom(\mathcal{K})$. For $\Delta = (Y_1 : Prop^{T_1, U_1}, \dots, Y_m : Prop^{T_m, U_m})$, we write $\lambda\Delta.\varphi$ for $\lambda Y_1 \dots \lambda Y_m.\varphi$ and $\Delta \rightarrow \kappa$ for $Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow \kappa$.

Theorem 5.8. *Let ϕ be a PHFL formula such that $\mathcal{K}; \Delta \vdash_M \phi : \kappa$. Then, for every $\rho \in \llbracket \mathcal{K} \rrbracket$, we have $\llbracket tr(\mathcal{K}) \vdash \lambda\Delta.\phi : tr(\Delta \rightarrow \kappa) \rrbracket(\rho) \in \llbracket \Delta \rightarrow \kappa \rrbracket$.*

Proof. By induction on the structure of derivation $\mathcal{K}; \Delta \vdash_M \phi : \kappa$, with case analysis on the last rule used. We use Lemma 5.7 for the cases of T-MU and T-NU. For the cases of T-CONJ and T-DISJ, the condition $T_1 \cup U_1 \cup T_2 \cup U_2 = S_M$ plays the key role. Note that if s belongs to $T_i \cup U_i$, then the semantics of $\lambda\Delta.\phi_0 \wedge \phi_1$ at s coincides with either that of $\lambda\Delta.\phi_{1-i}$ or a constant function $\lambda\tilde{v}.0$. Other cases are easy. \square

Let us discuss another important property of the type system. For $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U}$ and $f, g \in \llbracket tr(\kappa) \rrbracket$, we write $f \lesssim_\kappa g$ if $f v_1 \dots v_k \sqsubseteq_{Prop} g v_1 \dots v_k$ for every $(v_1, \dots, v_k) \in \llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_k, U_k} \rrbracket$. For interpretations $\rho_0, \rho_1 \in \llbracket \mathcal{K} \rrbracket$, we write $\rho_0 \lesssim_{\mathcal{K}} \rho_1$ if $\rho_0(X) \lesssim_{\mathcal{K}(X)} \rho_1(X)$ for every $X \in dom(\mathcal{K})$. We write $f \sim_\kappa g$ for $f \preceq_\kappa g \wedge g \preceq_\kappa f$, and analogously for interpretations. The results in the rest of this subsection allows us to compute the semantics of a formula of type κ up to \sim_κ .

Lemma 5.9. *Let $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U}$ be a refined type and h_0 and h_1 be monotone functions on $tr(\kappa)$ such that $x \in \llbracket \kappa \rrbracket$ implies $h_0 x, h_1 x \in \llbracket \kappa \rrbracket$ and that $x_0 \lesssim_\kappa x_1$ implies $h_0 x_0 \lesssim_\kappa h_1 x_1$.*

- If $U = \emptyset$, then $LFP(h_0) \lesssim_\kappa LFP(h_1)$.
- If $T = \emptyset$, then $GFP(h_0) \lesssim_\kappa GFP(h_1)$.

Here LFP and GFP are taken in $\llbracket tr(\kappa) \rrbracket$.

Proof. We prove the former; the proof of the latter is analogous. As discussed in the proof of Lemma 5.7, $LFP(h_0) = h_0^\alpha(\perp)$ and $LFP(h_1) = h_1^\alpha(\perp)$ for some sufficiently large ordinal α . We prove $h_0^\alpha(\perp) \lesssim_\kappa h_1^\alpha(\perp)$ by induction on α . Trivially $h_0^\alpha(\perp) = \perp \lesssim_\kappa \perp = h_1^\alpha(\perp)$. For $\alpha = \beta + 1$, since $h_0^\beta(\perp) \lesssim_\kappa h_1^\beta(\perp)$ by the induction hypothesis, $h_0(h_0^\beta(\perp)) \lesssim_\kappa h_1(h_1^\beta(\perp))$ follows from the assumption. If α is a limit cardinal, then $h_i^\alpha = \bigsqcup_{\beta < \alpha} h_i^\beta$. For every $(v_1, \dots, v_k) \in \llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_k, U_k} \rrbracket$, we have $h_i^\alpha(\perp)(v_1) \dots (v_k) = \bigsqcup_{\beta < \alpha} (h_i^\beta(\perp)(v_1) \dots (v_k))$ by

definition. Since $h_0^\beta(\perp)(v_1) \dots (v_k) \lesssim_\kappa h_1^\beta(\perp)(v_1) \dots (v_k)$ holds for every $\beta < \alpha$ by the induction hypothesis, we have $h_0^\alpha(\perp)(v_1) \dots (v_k) \lesssim_\kappa h_1^\alpha(\perp)(v_1) \dots (v_k)$. \square

Lemma 5.10. *Let ϕ be a PHFL formula such that $\mathcal{K}; \Delta \vdash_M \phi : \kappa$. Then, for every $\rho, \rho' \in \llbracket \mathcal{K} \rrbracket$ such that $\rho \lesssim_{\mathcal{K}} \rho'$, we have $\llbracket \lambda \Delta. \phi \rrbracket(\rho) \lesssim_{\Delta \rightarrow \kappa} \llbracket \lambda \Delta. \phi \rrbracket(\rho')$.*

Proof. By induction on the structure of derivation $\mathcal{K}; \Delta \vdash_M \phi : \kappa$. The most cases are easy. For fixpoints, we appeal to Lemma 5.9. The details are given in Appendix A. \square

The above lemma shows that the semantic domain for κ can be regarded as $\llbracket \kappa \rrbracket$ modulo \sim_κ . Furthermore the least and greatest fixpoints can be characterized in terms of the preorder \lesssim_κ .

Lemma 5.11. *Let κ be a refined type and h be a monotone function on $tr(\kappa)$ such that $x \in \llbracket \kappa \rrbracket$ implies $hx \in \llbracket \kappa \rrbracket$ and that $x \lesssim_\kappa x'$ implies $hx \lesssim_\kappa hx'$. If $LFP(h) \in \llbracket \kappa \rrbracket$, then $LFP(h)$ is a least element in $\{x \in \llbracket \kappa \rrbracket \mid hx \lesssim_\kappa x\}$ with respect to \lesssim_κ . Similarly, if $GFP(h) \in \llbracket \kappa \rrbracket$, then $GFP(h)$ is a greatest postfixpoint of h w.r.t \lesssim_κ .*

Proof. Assume $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_k, U_k} \rightarrow Prop^{T, U}$. Let \perp_κ be the least element of $\llbracket \kappa \rrbracket$, which is defined by: for every $s \in S_M$,

$$\perp_{Prop^{T, U}}(s) = \begin{cases} 1 & \text{if } s \in U \\ 0 & \text{otherwise.} \end{cases}$$

$$\perp_\kappa v_1 \dots v_k = \begin{cases} \perp_{Prop^{T, U}} & \text{if } v_i \sqsupseteq \perp_{Prop^{T_i, U_i}} \text{ for every } i \\ \lambda s. 0 & \text{otherwise.} \end{cases}$$

For an ordinal number α , we define x_α by $x_0 = \perp_\kappa$, $x_{\beta+1} = h(x_\beta)$ and $x_\alpha = \bigsqcup_{\beta < \alpha} x_\beta$ if α is a limit ordinal. By the monotonicity of h and Lemma 5.6, the sequence $\{x_\alpha\}_\alpha$ is well-defined and forms an increasing chain. Thus, there exists an ordinal α such that x_α is a fixpoint of h , and in particular, an element of $\{x \in \llbracket \kappa \rrbracket \mid hx \lesssim_\kappa x\}$. It follows by straightforward induction on α that x_α is also a least element of $\{x \in \llbracket \kappa \rrbracket \mid hx \lesssim_\kappa x\}$. Now, by the assumption that $LFP(h) \in \llbracket \kappa \rrbracket$, $LFP(h)$ is also an element of $\{x \in \llbracket \kappa \rrbracket \mid hx \lesssim_\kappa x\}$, and since $LFP(h) \sqsubseteq x_\alpha$, $LFP(h)$ is also a least element of $\{x \in \llbracket \kappa \rrbracket \mid hx \lesssim_\kappa x\}$. The proof for $GFP(h)$ is analogous. \square

If $\mathcal{K}; \emptyset \vdash_M \mu X. \phi : \kappa$ and $\rho \in \llbracket \mathcal{K} \rrbracket$, then $h = \llbracket \lambda X. \phi \rrbracket(\rho)$ satisfies the condition of the lemma above (recall Lemma 5.10). Hence $\llbracket \mu X. \phi \rrbracket(\rho)$ is \sim_κ -equivalent to every least fixpoint of $\llbracket \lambda X. \phi \rrbracket(\rho)$ with respect to \lesssim_κ ; in other words, to compute $\llbracket \mu X. \phi \rrbracket(\rho)$ up to \sim_κ , it suffices to compute a least fixpoint of $\llbracket \lambda X. \phi \rrbracket(\rho)$ in the quotient set $\llbracket \kappa \rrbracket / \sim_\kappa$. A similar statement holds for $\nu X. \phi$.

5.2. Decidability. This subsection gives a description of the interpretation of a PHFL formula using the first-order theory of reals, and obtains the decidability of the restricted fragment of PHFL model checking. To this end, we need to represent an element in $\llbracket \kappa \rrbracket$ as a tuple of reals. Each element of $\llbracket Prop^{T, U} \rrbracket$ can be naturally written as an n -tuple of $[0, 1]$ where n is the number of states in M . What remains is a representation of functions $\llbracket Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow Prop^{T, U} \rrbracket$.

The key result is Lemma 5.10, which allows us to identify $x, y \in \llbracket \kappa \rrbracket$ if $x \sim_\kappa y$. Let $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow Prop^{T, U}$. Since $x \in \llbracket \kappa \rrbracket$ is affine on $\llbracket Prop^{T_1, U_1} \rrbracket \times$

$\dots \times \llbracket Prop^{T_m, U_m} \rrbracket$, the \sim_κ -equivalence class of x contains an affine function on $[0, 1]^m$, i.e., $x \sim_\kappa f$ holds for some

$$f v_1 \dots v_m s = c_s + \sum_{i=1}^m \sum_{s' \in S} c_{s, i, s'} v_i(s').$$

We can use the tuple of coefficients $(c_s, (c_{s, i, s'})_{i \leq k, s' \in S})_{s \in S}$ as a representation of x .

Henceforth, we assume the set S_M of states of M is $\{1, \dots, n\}$. We define the *affine semantics* $(\llbracket Prop^m \rightarrow Prop \rrbracket)$ of type $Prop^m \rightarrow Prop$ by

$$\begin{aligned} (\llbracket Prop^m \rightarrow Prop \rrbracket) = \{ & (c_{i, j, k})_{i \in \{1, \dots, n\}, j \in \{0, \dots, m\}, k \in \{0, \dots, n\}} \in [0, 1]^{n(m+1)(n+1)} \mid \\ & \sum_{j=0}^m \sum_{k=0}^n c_{i, j, k} \leq 1 \text{ for every } i = 1, \dots, n. \} \end{aligned}$$

The tuple $(c_{i, j, k})_{i, j, k} \in (\llbracket \kappa \rrbracket)$ represents the function $f \in \llbracket Prop^m \rightarrow Prop \rrbracket$ such that, for each $i \in S_M = \{1, \dots, n\}$,

$$f g_1 \dots g_m i = c_{i, 0, 0} + \sum_{j \in \{1, \dots, m\}, k \in \{1, \dots, n\}} c_{i, j, k} \cdot g_j(k).$$

We write $(c_{i, j, k})_{i, j, k}^\dagger$ for the function f above. The *affine semantics* $(\llbracket \kappa \rrbracket)$ of refined type κ is defined by

$$(\llbracket \kappa \rrbracket) = \{ (c_{i, j, k})_{i, j, k} \in (\llbracket tr(\kappa) \rrbracket) \mid (c_{i, j, k})_{i, j, k}^\dagger \in \llbracket \kappa \rrbracket \}.$$

Lemma 5.12. *For every $x \in \llbracket \kappa \rrbracket$, there exists $\mathbf{c} = (c_{i, j, k})_{i, j, k} \in (\llbracket \kappa \rrbracket)$ such that $x \sim_\kappa \mathbf{c}^\dagger$.*

Proof. Let $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow Prop^{T, U}$. Since $x \in \llbracket \kappa \rrbracket$, it is affine on $\llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_m, U_m} \rrbracket$, i.e. for every $(v_1, \dots, v_m) \in \llbracket Prop^{T_1, U_1} \rrbracket \times \dots \times \llbracket Prop^{T_m, U_m} \rrbracket$ and $i \in S_M$,

$$x v_1 \dots v_m i = c_{i, 0, 0} + \sum_{j=1}^m \sum_{k=1}^n c_{i, j, k} v_j(k).$$

We can assume without loss of generality that $c_{i, j, k} = 0$ if $k \in T_j$ since $v_j(k) = 0$ in this case.

Let $c_{i, 0, k} = c_{i, j, 0} = 0$ for $j, k \geq 0$. Then $(c_{i, j, k})_{i, j, k} \in (\llbracket Prop^m \rightarrow Prop \rrbracket)$ and $(c_{i, j, k})_{i, j, k}^\dagger \sim_\kappa x$. Therefore $(c_{i, j, k})_{i, j, k} \in (\llbracket \kappa \rrbracket)$ since the latter implies $(c_{i, j, k})_{i, j, k}^\dagger \in \llbracket \kappa \rrbracket$. \square

Remark 5.13. For $x \in \llbracket \kappa \rrbracket$, $\mathbf{c} \in (\llbracket \kappa \rrbracket)$ such that $x \sim_\kappa \mathbf{c}^\dagger$ is not necessarily unique. The representation becomes unique if one imposes the following conditions. Assume that $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow Prop^{T, U}$.

- $c_{i, j, 0} = c_{i, 0, k} = 0$ for $j, k > 0$. Note that these coefficients are not used in $(c_{i, j, k})_{i, j, k}^\dagger$.
- $c_{i, j, k} = 0$ if $k \in T_j$. If $k \in T_j$, then $v_j(k) = 0$ and thus $c_{i, j, k}$ does not affect the value of $(c_{i, j, k})_{i, j, k}^\dagger$. So we can assume without loss of generality that $c_{i, j, k} = 0$.
- $c_{i, j, k} = 0$ if $k \in U_j$. If $k \in U_j$, then $v_j(k) = 1$. So, by adding $c_{i, j, k}$ to the constant part $c_{i, 0, 0}$ if necessarily, we obtain another representation that belongs to the same \sim_κ -equivalence class and that $c_{i, j, k} = 0$.

A representation $\mathbf{c} \in (\llbracket \kappa \rrbracket)$ is *canonical* if it satisfies the above conditions. It is not difficult to see that each \sim_κ -equivalence class has exactly one canonical representation.

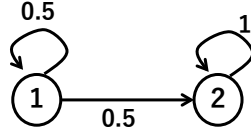
The second and third conditions are convenient for computing the affine semantics of $\phi_1 \vee \phi_2$ and $\phi_1 \wedge \phi_2$. Suppose $\emptyset; \Delta \vdash_M \phi_i : Prop^{T_i, U_i}$ for $i \in \{1, 2\}$ and $T_1 \cup U_1 \cup T_2 \cup U_2 = S_M$. If $(c_{i, j, k})_{i, j, k}$ and $(c'_{i, j, k})_{i, j, k}$ are canonical representations for the (affine) semantics of $\lambda \Delta. \phi_1$

and $\lambda\Delta.\phi_2$ respectively, then the semantics $(c''_{i,j,k})_{i,j,k}$ of $\lambda\Delta.\phi_1 \wedge \phi_2$ is also obtained pointwise by:

$$c''_{i,j,k} = \begin{cases} \min(c_{i,0,0}, c'_{i,0,0}) & \text{if } j = k = 0 \\ c'_{i,j,k} & \text{if } i \in U_1 \\ c_{i,j,k} & \text{if } i \in U_2 \setminus U_1 \\ 0 & \text{otherwise} \end{cases} \quad \square$$

As stated before, we will describe the affine semantics of a well-typed PHFL formula using the first-order theory of reals. Before doing so, however, we show an example of directly computing the affine semantics.

Example 5.14. Let M be the Markov chain $(\{1, 2\}, P, \rho_{AP}, 1)$ where $\rho_{AP}(p_i) = \{i\}$ for $i \in \{1, 2\}$, and $P(1, 1) = P(1, 2) = 0.5$ and $P(2, 2) = 1$, as depicted below.



Let ϕ be $\mu X.\lambda Y.(p_2 \wedge Y) \vee \bigcirc(p_1 \wedge X(Y))$. For $\rho \in \llbracket \mathcal{K} \rrbracket$, we write $\llbracket \mathcal{K}; \Delta \vdash_M \phi : \kappa \rrbracket(\rho)$ for the (canonical) matrix representation of $\llbracket \text{tr}(\mathcal{K}) \vdash \lambda \text{tr}(\Delta).\phi \rrbracket(\rho^\dagger)$. Let us compute $\llbracket \emptyset; \emptyset \vdash_M \phi : \kappa \rrbracket$ where $\kappa = \text{Prop}^{\emptyset, \emptyset} \rightarrow \text{Prop}^{\emptyset, \emptyset}$.

Let ρ be $\{X \mapsto (v_{i,j,k})_{i \in \{1,2\}, j \in \{0,1\}, k \in \{0,1,2\}}\}$. We write below an element $(c_{i,j,k})_{i,j,k}$ of $\llbracket \kappa \rrbracket$ as a matrix $(M_1 \ M_2)$, where M_i denotes $(c_{i,j,k})_{j,k}$. We can compute the affine semantics of subexpressions as follows.

$$\begin{aligned} \llbracket X : \kappa; Y : \text{Prop}^{\emptyset, \emptyset} \vdash_M Y : \text{Prop}^{\emptyset, \emptyset} \rrbracket(\rho) &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \\ \llbracket X : \kappa; Y : \text{Prop}^{\emptyset, \emptyset} \vdash_M p_2 : \text{Prop}^{\{1\}, \{2\}} \rrbracket(\rho) &= \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\ \llbracket X : \kappa; Y : \text{Prop}^{\emptyset, \emptyset} \vdash_M p_2 \wedge Y : \text{Prop}^{\{1\}, \emptyset} \rrbracket(\rho) &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

(Recall the discussion in the latter half of Remark 5.13.)

$$\llbracket X : \kappa; Y : \text{Prop}^{\emptyset, \emptyset} \vdash_M X(Y) : \text{Prop}^{\emptyset, \emptyset} \rrbracket(\rho) = \begin{pmatrix} v_{1,0,0} & 0 & 0 & v_{2,0,0} & 0 & 0 \\ 0 & v_{1,1,1} & v_{1,1,2} & 0 & v_{2,1,1} & v_{2,1,2} \end{pmatrix}$$

$$\begin{aligned} \llbracket X : \kappa; Y : \text{Prop}^{\emptyset, \emptyset} \vdash_M \bigcirc X(Y) : \text{Prop}^{\emptyset, \emptyset} \rrbracket(\rho) \\ = \begin{pmatrix} \frac{1}{2}(v_{1,0,0} + v_{2,0,0}) & 0 & 0 & v_{2,0,0} & 0 & 0 \\ 0 & \frac{1}{2}(v_{1,1,1} + v_{2,1,1}) & \frac{1}{2}(v_{1,1,2} + v_{2,1,2}) & 0 & v_{2,1,1} & v_{2,1,2} \end{pmatrix} \end{aligned}$$

(\bigcirc can be computed pointwise)

$$\llbracket X : \kappa; Y : \text{Prop}^{\emptyset, \emptyset} \vdash_M p_1 : \text{Prop}^{\{2\}, \{1\}} \rrbracket(\rho) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \llbracket X : \kappa; Y : \text{Prop}^{\emptyset, \emptyset} \vdash_M p_1 \wedge \bigcirc X(Y) : \text{Prop}^{\{2\}, \emptyset} \rrbracket(\rho) \\ = \begin{pmatrix} \frac{1}{2}(v_{1,0,0} + v_{2,0,0}) & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(v_{1,1,1} + v_{2,1,1}) & \frac{1}{2}(v_{1,1,2} + v_{2,1,2}) & 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

(Remark 5.13.)

$$\begin{aligned}
& (\!| X : \kappa; Y : Prop^{\emptyset, \emptyset} \vdash_M (p_2 \wedge Y) \vee (p_1 \wedge \bigcirc X(Y)) : Prop^{\emptyset, \emptyset} \!|)(\rho) \\
&= \begin{pmatrix} \frac{1}{2}(v_{1,0,0} + v_{2,0,0}) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(v_{1,1,1} + v_{2,1,1}) & \frac{1}{2}(v_{1,1,2} + v_{2,1,2}) & 0 & 0 & 0 & 1 \end{pmatrix} \\
&\quad \text{(Analogous to the computation of } \wedge \text{ discussed in Remark 5.13.)} \\
& (\!| X : \kappa; \emptyset \vdash_M \lambda Y.(p_2 \wedge Y) \vee (p_1 \wedge \bigcirc X(Y)) : \kappa \!|)(\rho) \\
&= \begin{pmatrix} \frac{1}{2}(v_{1,0,0} + v_{2,0,0}) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(v_{1,1,1} + v_{2,1,1}) & \frac{1}{2}(v_{1,1,2} + v_{2,1,2}) & 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

Thus, we have:

$$\begin{aligned}
& (\!| \emptyset; \emptyset \vdash_M \phi : \kappa \!|)(\emptyset) \\
&= LFP(\lambda \begin{pmatrix} v_{1,0,0} & - & - & v_{2,0,0} & - & - \\ - & v_{1,1,1} & v_{1,1,2} & - & v_{2,1,1} & v_{2,1,2} \end{pmatrix} \cdot \\
&\quad \begin{pmatrix} \frac{1}{2}(v_{1,0,0} + v_{2,0,0}) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(v_{1,1,1} + v_{2,1,1}) & \frac{1}{2}(v_{1,1,2} + v_{2,1,2}) & 0 & 0 & 0 & 1 \end{pmatrix}) \\
&= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}.
\end{aligned}$$

Using the result above, we obtain:

$$(\!| \emptyset; \emptyset \vdash_M \phi(p) : \kappa \!|)(\emptyset) = (1 \ 0 \ 0 \ 1 \ 0 \ 0).$$

Thus, $\phi(p_2)$ holds at state 1 with probability 1. \square

In the example above, we have directly computed the affine semantics. In general, however, we describe the affine semantics $(c_{i,j,k})_{i,j,k}$ by using logical formulas, to deal with arbitrary alternations of fixpoint operators. All the operations and properties on $\llbracket \kappa \rrbracket$ required for computing the affine semantics are definable by using the first-order theory of reals. Assume $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow Prop^{T, U}$ and let $\mathbf{c} = (c_{i,j,k})_{i,j,k} \in (\!| \kappa \!|)$. For example, the value $\mathbf{c}^\dagger v_1 \dots v_m i$ can be represented as a term

$$c_{i,0,0} + \sum_{j=1}^m \sum_{k=1}^n c_{i,j,k} v_j(k).$$

Then, for $\mathbf{c}, \mathbf{d} \in (\!| \kappa \!|)$, the relation $\mathbf{c} \lesssim_\kappa \mathbf{d}$ is written as

$$\begin{aligned}
& \forall v_1, \dots, v_m \in [0, 1]^n. \\
& \left(\bigwedge_{j=1, \dots, m} v_j \in (\!| Prop^{T_j, U_j} \!|) \right) \Rightarrow \bigwedge_{i=1, \dots, n} \left[(\mathbf{c}^\dagger v_1 \dots v_m i) \leq (\mathbf{d}^\dagger v_1 \dots v_m i) \right]
\end{aligned}$$

and $\mathbf{c}^\dagger \sim_\kappa \mathbf{d}^\dagger$ as

$$\mathbf{c}^\dagger \lesssim_\kappa \mathbf{d}^\dagger \wedge \mathbf{d}^\dagger \lesssim_\kappa \mathbf{c}^\dagger.$$

The meets and joins are also describable: for example, $\mathbf{c}_1^\dagger \sqcap \mathbf{c}_2^\dagger \sim_\kappa \mathbf{d}^\dagger$ is equivalent to

$$\begin{aligned}
& \forall v_1, \dots, v_m \in [0, 1]^n. \left(\bigwedge_{j=1, \dots, m} v_j \in (\!| Prop^{T_j, U_j} \!|) \right) \\
& \Rightarrow \bigwedge_{i=1, \dots, n} \left[(\mathbf{d}^\dagger v_1 \dots v_m i) = \max \left((\mathbf{c}_1^\dagger v_1 \dots v_m i), (\mathbf{c}_2^\dagger v_1 \dots v_m i) \right) \right];
\end{aligned}$$

note that max as well as min is an operation definable in the first-order theory of reals.

Lemma 5.15. *Suppose $\mathcal{K}; \Delta \vdash_M \phi : \kappa$ and $\mathcal{K} = \{X_1 : \kappa_1, \dots, X_N : \kappa_N\}$. Then one can effectively construct a formula Φ of the first-order real arithmetic such that, for every $\mathbf{c}_\ell \in (\downarrow \kappa_\ell)$ ($\ell = 1, \dots, N$) and $\mathbf{d} \in (\downarrow \Delta \rightarrow \kappa)$,*

$$\Phi(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d}) \Leftrightarrow \llbracket \lambda \Delta. \phi \rrbracket (\{X_\ell \mapsto \mathbf{c}_\ell^\dagger \mid 1 \leq \ell \leq N\}) \sim_{\Delta \rightarrow \kappa} \mathbf{d}^\dagger.$$

Proof. By induction on the structure of derivation $\mathcal{K}; \Delta \vdash_M \phi : \kappa$. The most cases are easy. For example, consider the case that the last rule is (T-CONJ). Then $\phi = \phi_1 \wedge \phi_2$ and $\mathcal{K}; \Delta \vdash_M \phi_i : Prop^{T_i, U_i}$ for $i = 1, 2$. By the induction hypothesis, we have predicates Φ_1 and Φ_2 representing $\llbracket \lambda \Delta. \phi_1 \rrbracket$ and $\llbracket \lambda \Delta. \phi_2 \rrbracket$. Then $\Phi(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d})$ is defined as

$$\begin{aligned} \exists \mathbf{d}_1 \in (\downarrow \Delta \rightarrow Prop^{T_1, U_1}) . \exists \mathbf{d}_2 \in (\downarrow \Delta \rightarrow Prop^{T_2, U_2}) . \\ \Phi_1(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d}_1) \wedge \Phi_2(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d}_2) \wedge \mathbf{d}^\dagger \sim_\kappa \mathbf{d}_1^\dagger \sqcap \mathbf{d}_2^\dagger. \end{aligned}$$

Since $\llbracket \lambda \Delta. \phi_i \rrbracket (\{X_\ell \mapsto \mathbf{c}_\ell^\dagger \mid 1 \leq \ell \leq N\}) \in \llbracket \Delta \rightarrow Prop^{T_i, U_i} \rrbracket$ by Theorem 5.8, there exists a $\mathbf{d}_i \in (\downarrow \Delta \rightarrow Prop^{T_i, U_i})$ such that $\Phi_i(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d}_i)$.

The only non-trivial cases are fixpoints. Consider the case that the last rule is (T-MU). Then $\phi = \nu X. \phi_0$ and $\mathcal{K}, X : \kappa; \Delta \vdash_M \phi_0 : \kappa$. By the induction hypothesis, we have Φ_0 representing the semantics of ϕ_0 . Then $\Phi(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d})$ is defined as

$$\Phi_0(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d}, \mathbf{d}) \wedge \forall \mathbf{d}' \in (\downarrow \kappa) . \Phi_0(\mathbf{c}_1, \dots, \mathbf{c}_N, \mathbf{d}', \mathbf{d}') \Rightarrow \mathbf{d}^\dagger \preceq_\kappa \mathbf{d}'^\dagger.$$

The first condition says that \mathbf{d}^\dagger is a fixpoint of $\llbracket \lambda X. \phi_0 \rrbracket$ modulo \sim_κ . The second condition says that \mathbf{d}^\dagger is a least element in the set of fixpoints \mathbf{d}'^\dagger of $\llbracket \lambda X. \phi_0 \rrbracket$ modulo \sim_κ . Correctness of the above formula follows from Corollary 5.11. \square

Theorem 5.16. *Given a closed formula ϕ and a Markov chain M such that $\emptyset; \emptyset \vdash_M \phi : Prop^{T, U}$, it is decidable whether $M \models \phi$.*

Proof. Assume that $s_{in, M} = 1$. By Lemma 5.15, one can effectively construct a formula Φ of the first-order real arithmetic such that $\Phi(r_1, \dots, r_n)$ is valid if and only if $\llbracket \phi \rrbracket_M(\emptyset)(i) = r_i$ for every $i = 1, \dots, n$. Hence $M \models \phi$ if and only if $\exists r_2, \dots, r_n. \Phi(1, r_2, \dots, r_n)$. The validity of this formula is decidable [Tar51]. \square

Remark 5.17. The formula obtained in the proof above contains both universal and existential quantifiers in general; hence the complexity is doubly exponential time in general [DH88]. However, if ϕ is of the form $[\phi']_{\geq r}$ where ϕ' contains no occurrences of ν and $[\cdot]_J$, then we can express Φ using only existential quantifiers, and appeal to a decision algorithm for the existential theory of the reals, whose complexity is PSPACE [Can88]. In fact, to deal with μ -formulas in the restricted fragment, it suffices to consider only inequalities of the form $LFP(h) \preceq_\kappa x$, which can be represented by an existential formula $\exists x'. (x' \preceq_\kappa x \wedge h x' \preceq_\kappa x')$. Note that \preceq is definable by a quantifier-free formula, as stated in the lemma below. Note also that the semantics of conjunctions and disjunctions can be expressed without using quantifiers, as discussed in Remark 5.13.

Lemma 5.18. *Let $\kappa = Prop^{T_1, U_1} \rightarrow \dots \rightarrow Prop^{T_m, U_m} \rightarrow Prop^{T, U}$. For $J \subseteq \{1, \dots, m\} \times \{1, \dots, n\}$, we write $J \in \mathcal{J}(\kappa)$ if the characteristic function belongs to $\prod_{i=1}^m \llbracket Prop^{T_i, U_i} \rrbracket$, i.e.,*

- if $k \in T_j$, then $(k, j) \notin J$, and
- if $k \in U_j$, then $(k, j) \in J$.

Let $(c_{i,j,k})_{i,j,k}, (c'_{i,j,k})_{i,j,k} \in (\kappa)$. Then, $(c_{i,j,k})_{i,j,k}^\dagger \preceq_\kappa (c'_{i,j,k})_{i,j,k}^\dagger$ if and only if:

$$\bigwedge_{i \in \{1, \dots, n\}} \bigwedge_{J \in \mathcal{J}(\kappa)} \left(c_{i,0,0} + \sum_{(j,k) \in J} c_{i,j,k} \leq c'_{i,0,0} + \sum_{(j,k) \in J} c'_{i,j,k} \right).$$

Proof. **If:** Suppose $g_1 \in \llbracket Prop^{T_1, U_1} \rrbracket, \dots, g_m \in \llbracket Prop^{T_m, U_m} \rrbracket$. We regard $(g_1, \dots, g_m) = (g_j(k))_{j \in \{1, \dots, m\}, k \in \{1, \dots, n\}}$ as an mn -dimensional vector. We also identify $J \in \mathcal{J}(\kappa)$ with $(h_j(k))_{j,k}$ where $h_j(k) = 1$ if $(j, k) \in J$ and $h_j(k) = 0$ if $(j, k) \notin J$.

Let $0 = s_0 < s_1 < s_2 < \dots < s_N = 1$ be the sequence obtained by sorting $\{g_j(k) \mid 1 \leq j \leq m, 1 \leq k \leq n\} \cup \{0, 1\}$. For each $\ell > 0$, we define J_ℓ and r_ℓ by

$$J_\ell = \{(j, k) \mid s_\ell \leq h_j(k)\} \quad \text{and} \quad r_\ell = s_\ell - s_{\ell-1}.$$

Then

$$(g_1, \dots, g_m) = \sum_{\ell} r_\ell J_\ell.$$

Hence

$$\begin{aligned} (c_{i,j,k})_{i,j,k}^\dagger g_1 \dots g_m i &= \sum_{\ell} r_\ell (c_{i,0,0} + \sum_{(j,k) \in J_\ell} c_{i,j,k}) \\ &\leq \sum_{\ell} r_\ell (c'_{i,0,0} + \sum_{(j,k) \in J_\ell} c'_{i,j,k}) \\ &= (c'_{i,j,k})_{i,j,k}^\dagger g_1 \dots g_m i. \end{aligned}$$

Only if: For each $J \in \mathcal{J}(\kappa)$, let g_j be the element of $\llbracket Prop \rrbracket$ such that $g_j(k) = 1$ if $(j, k) \in J$ and $g_j(k) = 0$ if $(j, k) \notin J$. Then $g_j \in \llbracket Prop^{T_j, U_j} \rrbracket$ for every $j = 1, \dots, m$. We have

$$c_{i,0,0} + \sum_{(j,k) \in J} c_{i,j,k} = (c_{i,j,k})_{i,j,k}^\dagger g_1 \dots g_m i \leq (c'_{i,j,k})_{i,j,k}^\dagger g_1 \dots g_m i = c'_{i,0,0} + \sum_{(j,k) \in J} c'_{i,j,k},$$

as required. □

5.3. Expressivity. We have already seen in Example 5.3 and Lemma 5.5 that the decidable fragment of the order-1 PHFL model checking problem strictly subsumes μ^p -calculus model checking. To provide a further evidence of the expressivity of the decidable fragment, in this subsection, we show that the termination problem for recursive Markov chains (or, probabilistic pushdown systems) [EY09, BEKK13] can be encoded into the decidable fragment of order-1 PHFL model checking. Since recursive Markov chains are known to be equivalent to order-1 probabilistic higher-order recursion schemes (PHORS) [KLG20], we encode below the termination problem for the latter into the PHFL model checking problem.

We first recall the definition of PHORS, specialized for order 1.

Definition 5.19. An order-1 PHORS \mathcal{G} is a triple $(\mathcal{X}, \mathcal{R}, t)$, where:

- \mathcal{X} is a finite map from (order-1) variables to their arities;

- \mathcal{R} is a finite set of rules of the form:

$$X y_1 \cdots y_{\mathcal{X}(X)} \rightarrow t_L \oplus_p t_R,$$

where t_L, t_R range over the set of $(\mathcal{X}, \{y_1, \dots, y_{\mathcal{X}(X)}\})$ -terms. Here, the set of (\mathcal{X}, V) -terms, ranged over by t , is defined by the grammar:

$$t ::= \mathbf{e} \mid y \mid X t_1 \cdots t_{\mathcal{X}(X)},$$

where y and X range over V and $\text{dom}(\mathcal{X})$ respectively, and \mathbf{e} is a special symbol denoting termination.

- t is a (\mathcal{X}, \emptyset) -term.

For $\mathcal{G} = (\mathcal{X}_{\mathcal{G}}, \mathcal{R}_{\mathcal{G}}, t_{\mathcal{G}})$, the reduction relation $t \xrightarrow{\pi, p}_{\mathcal{G}} t'$ on terms (where $\pi \in \{L, R\}^*$ and $p \in [0, 1]$) is defined by:

$$\frac{}{t \xrightarrow{\epsilon, 1}_{\mathcal{G}} t}$$

$$\frac{t \xrightarrow{\pi, q}_{\mathcal{G}} X t_1 \cdots t_k \quad X y_1 \cdots y_k \rightarrow t_L \oplus_p t_R \in \mathcal{R}_{\mathcal{G}}}{t \xrightarrow{\pi L, q p}_{\mathcal{G}} [t_1/y_1, \dots, t_k/y_k] t_L}$$

$$\frac{t \xrightarrow{\pi, q}_{\mathcal{G}} X t_1 \cdots t_k \quad X y_1 \cdots y_k \rightarrow t_L \oplus_p t_R \in \mathcal{R}_{\mathcal{G}}}{t \xrightarrow{\pi R, q(1-p)}_{\mathcal{G}} [t_1/y_1, \dots, t_k/y_k] t_R}$$

Note that the reduction is deterministic: for $\pi \in \{L, R\}^*$, there exists at most one (p, t') such that $t \xrightarrow{\pi, p}_{\mathcal{G}} t'$. The *termination probability* of $\mathcal{G} = (\mathcal{X}_{\mathcal{G}}, \mathcal{R}_{\mathcal{G}}, t_{\mathcal{G}})$, written $\text{Prob}T(\mathcal{G})$, is defined by:

$$\text{Prob}T(\mathcal{G}, t, \pi) = \begin{cases} p & \text{if } t \xrightarrow{\pi, p}_{\mathcal{G}} \mathbf{e} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Prob}T(\mathcal{G}) = \sum_{\pi \in \{L, R\}^*} \text{Prob}T(\mathcal{G}, t_{\mathcal{G}}, \pi).$$

The following example has been taken from [KLG20].

Example 5.20. Let \mathcal{G} be the order-1 PHORS $(\mathcal{X}, \mathcal{R}, X_0)$, where:

$$\mathcal{X} = \{X_0 \mapsto 0, F \mapsto 1\}$$

$$\mathcal{R} = \{X_0 = X_1 \mathbf{e} \oplus_1 X_0, \quad X_1 y = y \oplus_p X_1(X_1 y)\}.$$

Then, we have $X_0 \xrightarrow{L, 1}_{\mathcal{G}} X_1 \mathbf{e} \xrightarrow{L, p}_{\mathcal{G}} \mathbf{e}$ and $X_0 \xrightarrow{L, 1}_{\mathcal{G}} X_1 \mathbf{e} \xrightarrow{R, 1-p}_{\mathcal{G}} X_1(X_1 \mathbf{e}) \xrightarrow{L, p}_{\mathcal{G}} X_1 \mathbf{e} \xrightarrow{L, p}_{\mathcal{G}} \mathbf{e}$. The termination probability $\text{Prob}T(\mathcal{G})$ is 1 if $p \geq \frac{1}{2}$ and $\frac{p}{1-p}$ if $p < \frac{1}{2}$. \square

We now encode an order-1 PHORS \mathcal{G} into a Markov chain $M_{\mathcal{G}}$ and a closed order-1 PHFL formula $\phi_{\mathcal{G}}$ such that $\emptyset; \emptyset \vdash_M \phi_{\mathcal{G}} : \text{Prop}^{\emptyset, \emptyset}$ and $\text{Prob}T(\mathcal{G})$ coincides with the probability that $\phi_{\mathcal{G}}$ holds at the initial state of $M_{\mathcal{G}}$ (i.e., $\text{Prob}T(\mathcal{G}) = \llbracket \phi_{\mathcal{G}} \rrbracket(\emptyset)(s_{\text{in}})$). We first give a construction of M . Let $\{p_1, \dots, p_m\}$ be the set $\{p, 1-p \mid \oplus_p \text{ occurs in } \mathcal{G}\}$. We assume $p_1 < p_2 < \dots < p_m$. We define the Markov chain $M_{\mathcal{G}} = (S, P, \rho_{AP}, s_{\text{in}})$ as follows.

- $S = \{s_0, s_1, \dots, s_{m+1}\}$,
- P satisfies $P(s_0, s_1) = p_1$, $P(s_0, s_i) = p_i - p_{i-1}$ for $2 \leq i \leq m$, $P(s_0, s_{m+1}) = 1 - p_m$, $P(s_i, s_0) = 1$ for $1 \leq i \leq m+1$ and $P(s_i, s_j) = 0$ otherwise,
- $\rho_{AP}(P_i) = \{s_i\}$ for each $0 \leq i \leq m+1$, and

- $s_{\text{in}} = s_0$.

Note that $\llbracket \bigcirc(P_1 \vee \dots \vee P_i) \rrbracket(\emptyset)(s_0) = p_i$ and $\llbracket \bigcirc(P_{i+1} \vee \dots \vee P_{m+1}) \rrbracket(\emptyset)(s_0) = 1 - p_i$. Now for each term of t of PHORS \mathcal{G} , we construct a formula $\langle t \rangle$, so that the termination probability of t coincides with the probability that ϕ_t holds at s_0 . The translation is given by:

$$\langle \mathbf{e} \rangle = \text{true} \quad \langle y \rangle = y \quad \langle X_i t_1 \dots t_k \rangle = X_i \langle t_1 \rangle \dots \langle t_k \rangle.$$

For each rule $X_i y_1 \dots y_{\mathcal{X}(X_i)} = t_L \oplus_{p_i} t_R$ of \mathcal{R} , we construct the following equality on PHFL formulas:

$$X_i = \lambda y_1, \dots, y_{\mathcal{X}(X_i)}. \bigcirc (((P_1 \vee \dots \vee P_i) \wedge \bigcirc \langle t_L \rangle) \vee ((P_{i+1} \vee \dots \vee P_{m+1}) \wedge \bigcirc \langle t_R \rangle)).$$

We thus obtain a system of mutually recursive equations $\{X_1 = \phi_1, \dots, X_k = \phi_k\}$, whose least solution θ (which maps each X_i to a formula ψ_i that satisfies $\psi_i = \theta\phi_i$) can be represented by using the least fixpoint operators in an obvious manner. We then let $\phi_{\mathcal{G}}$ be $\theta\langle t_{\mathcal{G}} \rangle$.

Example 5.21. Recall \mathcal{G} in Example 5.20. Assuming $p < \frac{1}{2}$, $M_{\mathcal{G}}$ is:

- $S = \{s_0, s_1, s_2, s_3, s_4\}$.
- $P(s_0, s_1) = p$, $P(s_0, s_2) = 1 - 2p$, $P(s_0, s_3) = p$, $P(s_0, s_4) = 0$, $P(s_i, s_0) = 1$ for $i \in \{1, 2, 3, 4\}$ and $P(s_i, s_j) = 0$ for $i, j > 0$.
- $\rho_{AP}(P_i) = \{s_i\}$ for each $i \in \{0, \dots, 4\}$, and
- $s_{\text{in}} = s_0$.

The rules of \mathcal{G} are translated to the following equations:

$$\begin{aligned} X_0 &= \bigcirc (((P_1 \vee P_2 \vee P_3) \wedge \bigcirc (X_1 \mathbf{e})) \vee (P_4 \wedge \bigcirc X_0)) \\ X_1 &= \lambda y. \bigcirc (((P_1 \wedge \bigcirc y) \vee ((P_2 \vee P_3 \vee P_4) \wedge \bigcirc (X_1(X_1 y)))) \end{aligned}$$

Thus, $\phi_{\mathcal{G}}$ is given as:

$$\mu X_0. \bigcirc (((P_1 \vee P_2 \vee P_3) \wedge \bigcirc (\psi_1 \mathbf{e})) \vee (P_4 \wedge \bigcirc X_0)),$$

where ψ_1 is:

$$\mu X_1. \lambda y. \bigcirc (((P_1 \wedge \bigcirc y) \vee ((P_2 \vee P_3 \vee P_4) \wedge \bigcirc (X_1(X_1 y))))).$$

Actually, $\phi_{\mathcal{G}}$ can be simplified to $\psi_1 \mathbf{e}$ in this case. \square

To see the correctness of the above encoding, recall that $\llbracket \bigcirc(P_1 \vee \dots \vee P_i) \rrbracket(\emptyset)(s_0) = p_i$ and $\llbracket \bigcirc(P_{i+1} \vee \dots \vee P_{m+1}) \rrbracket(\emptyset)(s_0) = 1 - p_i$. Thus, by the equality on X_i above, the probability that $\theta(X_i \langle t_1 \rangle \dots, \langle t_{\mathcal{X}(X_i)} \rangle)$ holds at s_0 is equivalent to $p_i \cdot q_L + (1 - p_i) \cdot q_R$, where q_L and q_R are respectively the probabilities that $\theta(\langle [t_1/y_1, \dots, t_{\mathcal{X}(X_i)}/y_{\mathcal{X}(X_i)}] t_L \rangle)$ and $\theta(\langle [t_1/y_1, \dots, t_{\mathcal{X}(X_i)}/y_{\mathcal{X}(X_i)}] t_R \rangle)$ hold at s_0 . Thus, the formula $\theta(X_i \langle t_1 \rangle \dots, \langle t_{\mathcal{X}(X_i)} \rangle)$ mimics the termination probability of $X_i \langle t_1 \rangle \dots, \langle t_{\mathcal{X}(X_i)} \rangle$, which is equivalent to $p_i \cdot q'_L + (1 - p_i) \cdot q'_R$, where q'_L and q'_R are respectively the termination probabilities of $[t_1/y_1, \dots, t_{\mathcal{X}(X_i)}/y_{\mathcal{X}(X_i)}] t_L$ and $[t_1/y_1, \dots, t_{\mathcal{X}(X_i)}/y_{\mathcal{X}(X_i)}] t_R$. We omit a formal proof of correctness of the encoding.

We now check that $\phi_{\mathcal{G}}$ belongs to the restricted fragment. To this end, it suffices to check that, for each rule $X y_1 \dots y_k \rightarrow t_L \oplus_p t_R \in \mathcal{R}$, the type judgment:

$$\mathcal{K}_{\mathcal{X}}; \Delta \vdash_{M_{\mathcal{G}}} \bigcirc (((P_1 \vee \dots \vee P_i) \wedge \bigcirc \langle t_L \rangle) \vee ((P_{i+1} \vee \dots \vee P_{m+1}) \wedge \bigcirc \langle t_R \rangle)) : Prop^{\emptyset, \emptyset}$$

holds for $\mathcal{K}_{\mathcal{X}} = \{X : (Prop^{\emptyset, \emptyset})^{\mathcal{X}(X)} \rightarrow Prop^{\emptyset, \emptyset} \mid X \in \text{dom}(\mathcal{X})\}$ and $\Delta = y_1 : Prop^{\emptyset, \emptyset}, \dots, y_k : Prop^{\emptyset, \emptyset}$.

By a straightforward induction on the structure of t_L , we have: $\mathcal{K}_{\mathcal{X}}; \Delta \vdash_{M_G} \langle t_L \rangle : Prop^{\emptyset, \emptyset}$, which implies $\mathcal{K}_{\mathcal{X}}; \Delta \vdash_{M_G} \bigcirc \langle t_L \rangle : Prop^{\emptyset, \emptyset}$. We also have:

$$\mathcal{K}_{\mathcal{X}}; \Delta \vdash_{M_G} P_1 \vee \cdots \vee P_i : Prop^{\{P_0, P_{i+1}, \dots, P_{m+1}\}, \{P_1, \dots, P_i\}}.$$

Thus, by using T-AND, we have:

$$\mathcal{K}_{\mathcal{X}}; \Delta \vdash_{M_G} (P_1 \vee \cdots \vee P_i) \wedge \bigcirc \langle t_L \rangle : Prop^{\{P_0, P_{i+1}, \dots, P_{m+1}\}, \emptyset}.$$

Similarly, we have:

$$\mathcal{K}_{\mathcal{X}}; \Delta \vdash_{M_G} (P_{i+1} \vee \cdots \vee P_{m+1}) \wedge \bigcirc \langle t_R \rangle : Prop^{\{P_0, P_1, \dots, P_i\}, \emptyset}.$$

Thus, by using T-OR and T-AVG, we obtain

$$\mathcal{K}_{\mathcal{X}}; \Delta \vdash_{M_G} \bigcirc(((P_1 \vee \cdots \vee P_i) \wedge \bigcirc \langle t_L \rangle) \vee ((P_{i+1} \vee \cdots \vee P_{m+1}) \wedge \bigcirc \langle t_R \rangle)) : Prop^{\emptyset, \emptyset}$$

as required.

6. RELATED WORK

As mentioned in Section 1, PHFL can be regarded as a probabilistic extension of the higher-order fixpoint logic, and as a higher-order extension of the μ^p -calculus. We thus compare our work with previous studies on (non-probabilistic) higher-order fixpoint logic and those on (non-higher-order) probabilistic logics. As already mentioned, for (non-probabilistic) HFL, model checking of finite-state systems is known to be decidable [VV04], and k -EXPTIME complete [ALS07]. This is in a sharp contrast with our result that PHFL model checking is highly undecidable (both Π_1^1 -hard and Σ_1^1 -hard) even at order 1.

As for studies on probabilistic logics, besides the μ^p -calculus, there are other probabilistic extensions of the modal μ -calculus [MM97, HK97, MS13]. Łukasiewicz μ -calculus introduced by Mio and Simpson [MS13] is among the most expressive ones, which has, in addition to \wedge and \vee , another kind of conjunction (\odot) and disjunction (\oplus), called Łukasiewicz operations. To our knowledge, ours is the first *higher-order* and probabilistic extension of the modal μ -calculus. The decidable fragment of PHFL studied in Section 5 is strictly more expressive than the μ^p -calculus. We conjecture that the expressive power of PHFL is incomparable to that of Łukasiewicz μ -calculus. On the one hand, the property defined by the PHFL formula in the proof of Theorem 2.9 cannot be expressed in Łukasiewicz μ -calculus, hence PHFL is not subsumed by Łukasiewicz μ -calculus. On the other hand, Łukasiewicz operations do not seem expressible in PHFL. It would be interesting to study a higher-order extension of Łukasiewicz μ -calculus (in other words, an extension of PHFL with Łukasiewicz operations).

Recently, Kobayashi et al. [KLG19] introduced PHORS, a probabilistic extension of higher-order recursion schemes (HORS), which can also be viewed as a higher-order extension of recursive Markov chains (or probabilistic pushdown systems), and proved that the almost sure termination problem is undecidable. Although the problem setting is quite different (in our work, the *logic* is higher-order whereas the *system* to be verified is higher-order in their work), our encoding of the μ -arithmetic in Section 4 has been partially inspired by their undecidability proof; they also represented a natural number n as the probability $\frac{1}{2^n}$.

In Section 3, we have used the undecidability of the value-1 problem for probabilistic automata to prove the undecidability of PHFL model checking. Fijalkow et al. [FGKO15] studied a decidable subclass of probabilistic automata called leaktight automata. The idea of their restriction appears to be quite different from our type-based restriction of the PHFL model checking problem.

7. CONCLUSION

We have introduced PHFL, a probabilistic logic which can be regarded as both a probabilistic extension of HFL and a higher-order extension of the probabilistic logic μ^p -calculus. We have shown that the model-checking problem for PHFL for a finite Markov chain is undecidable for the μ -only and order-1 fragment. We have also shown that the model-checking problem for the full order-1 fragment of PHFL is Π_1^1 -hard and Σ_1^1 -hard. As positive results, we have introduced a decidable subclass of the PHFL model-checking problem, and showed that the termination problem of Recursive Markov Chains can be encoded in the subclass.

Finding an upper bound of the hardness of the PHFL model-checking problem is left for future work. It is also left for future work to find a larger decidable class of PHFL model-checking problems.

Acknowledgements. We would like to thank anonymous referees for useful comments. This work was supported by JSPS KAKENHI Grant Number JP15H05706 and JP20H00577, and JP20H05703.

REFERENCES

- [ALS07] Roland Axelsson, Martin Lange, and Rafal Somla. The complexity of model checking higher-order fixpoint logic. *Logical Methods in Computer Science*, 3(2), 2007.
- [BEKK13] Tomáš Brázdil, Javier Esparza, Stefan Kiefer, and Antonín Kucera. Analyzing probabilistic pushdown automata. *Formal Methods in System Design*, 43(2):124–163, 2013.
- [Can88] John F. Canny. Some algebraic and geometric computations in PSPACE. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 460–467. ACM, 1988.
- [CKP15] Pablo F. Castro, Cecilia Kilmurray, and Nir Piterman. Tractable probabilistic mu-calculus that expresses probabilistic temporal logics. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPICs*, pages 211–223. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
- [DH88] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *J. Symb. Comput.*, 5(1/2):29–35, 1988.
- [EY09] Kousha Etessami and Mihalis Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1):1:1–1:66, 2009.
- [FGKO15] Nathanaël Fijalkow, Hugo Gimbert, Edon Kelmendi, and Youssouf Oualhadj. Deciding the value 1 problem for probabilistic leaktight automata. *Logical Methods in Computer Science*, 11(2), 2015.
- [Fij17] Nathanaël Fijalkow. Undecidability results for probabilistic automata. *SIGLOG News*, 4(4):10–17, 2017.
- [GO10] Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In Samson Abramsky, Cyril Gavaille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 527–538. Springer, 2010.
- [Har86] David Harel. Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *J. ACM*, 33(1):224–248, 1986.
- [HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.
- [HK97] Michael Huth and Marta Z. Kwiatkowska. Quantitative analysis and model checking. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 111–122. IEEE Computer Society, 1997.

- [KLB17] Naoki Kobayashi, Étienne Lozes, and Florian Bruse. On the relationship between higher-order recursion schemes and higher-order fixpoint logic. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 246–259. ACM, 2017.
- [KLG19] Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. On the termination problem for probabilistic higher-order recursive programs. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–14. IEEE, 2019.
- [KLG20] Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. On the Termination Problem for Probabilistic Higher-Order Recursive Programs. *Logical Methods in Computer Science*, Volume 16, Issue 4, October 2020.
- [KNP11] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.
- [Koz83] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.
- [KTW18] Naoki Kobayashi, Takeshi Tsukada, and Keiichi Watanabe. Higher-order program verification via HFL model checking. In Amal Ahmed, editor, *Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10801 of *Lecture Notes in Computer Science*, pages 711–738. Springer, 2018.
- [Loz15] Étienne Lozes. A type-directed negation elimination. In Ralph Matthes and Matteo Mio, editors, *Proceedings Tenth International Workshop on Fixed Points in Computer Science, FICS 2015, Berlin, Germany, September 11-12, 2015*, volume 191 of *EPTCS*, pages 132–142, 2015.
- [Lub89] Robert S. Lubarsky. mu-definable sets of integers. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 343–352. IEEE Computer Society, 1989.
- [MKT20] Yo Mitani, Naoki Kobayashi, and Takeshi Tsukada. A probabilistic higher-order fixpoint logic. In Zena M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*, volume 167 of *LIPICs*, pages 19:1–19:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- [MM97] Carroll Morgan and Annabelle McIver. A probabilistic temporal calculus based on expectations. In *Proc. Formal Methods Pacific*, pages 4–22. Springer, 1997.
- [MS13] Matteo Mio and Alex Simpson. Lukasiewicz mu-calculus. In David Baelde and Arnaud Carayol, editors, *Proceedings Workshop on Fixed Points in Computer Science, FICS 2013, Turino, Italy, September 1st, 2013*, volume 126 of *EPTCS*, pages 87–104, 2013.
- [Paz71] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.
- [Rab63] Michael O Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.
- [Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.
- [VV04] Mahesh Viswanathan and Ramesh Viswanathan. A higher order modal fixed point logic. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings*, volume 3170 of *Lecture Notes in Computer Science*, pages 512–528. Springer, 2004.

APPENDIX A. PROOFS

This section proofs omitted in the main text.

A.1. Proof of Theorem 4.8. The following lemma states that the relation \sim is preserved by the least upper bound operation.

Lemma A.1. *Let I be a set and T be a type of μ -arithmetic. Assume families $\{v_i\}_{i \in I}$ and $\{u_i\}_{i \in I}$ of elements of $\llbracket T \rrbracket_\mu$ and $\llbracket \text{tr}(T) \rrbracket$, respectively, and suppose that $v_i \sim_T u_i$ for every $i \in I$. Then $\bigsqcup_{i \in I} v_i \sim_T \bigsqcup_{i \in I} u_i$.*

Proof. By induction on the structure of T . The base case $T = \Omega$ is obvious. Assume that $T = A \rightarrow T'$.

Let $x \in \llbracket A \rrbracket_\mu$ and $y \in \llbracket tr(A) \rrbracket$ and assume that $x \sim_A y$. For each $i \in I$, since $v_i \sim_T u_i$, we have $v_i x \sim_{T'} u_i y$. By the induction hypothesis,

$$\bigsqcup_{i \in I} (v_i x) \sim_{T'} \bigsqcup_{i \in I} (u_i y).$$

Since the order on functions are component-wise, we have

$$\left(\bigsqcup_{i \in I} v_i \right) x = \bigsqcup_{i \in I} (v_i x) \quad \text{and} \quad \left(\bigsqcup_{i \in I} u_i \right) y = \bigsqcup_{i \in I} (u_i y).$$

So

$$\left(\bigsqcup_{i \in I} v_i \right) x \sim_{T'} \left(\bigsqcup_{i \in I} u_i \right) y.$$

Since $x \sim_A y$ is arbitrary, we have:

$$\left(\bigsqcup_{i \in I} v_i \right) \sim_{A \rightarrow T'} \left(\bigsqcup_{i \in I} u_i \right). \quad \square$$

We are now ready to prove Theorem 4.8.

Proof of Theorem 4.8. We prove the theorem by induction on the structure of φ . In this proof, we omit the subscript M of $\llbracket - \rrbracket_M$ for simplicity.

- Case $\varphi = X$. In this case, $tr(\varphi) = X$ and $\Gamma(X) = A$. Thus, we have:

$$\llbracket \Gamma \vdash_\mu \varphi : A \rrbracket_\mu(\theta) = \theta(X) \sim_{\Gamma(X)} \rho(X) = \llbracket tr(\Gamma) \vdash tr(\varphi) : tr(A) \rrbracket(\rho),$$

as required.

- Case $\varphi = Z$. In this case, $tr(\varphi) = p'_0$ and $A = N$. We have

$$\llbracket \Gamma \vdash_\mu \varphi : A \rrbracket_\mu(\theta) = 0 \sim_N (1, 0, 0, 0) = \llbracket tr(\Gamma) \vdash tr(\varphi) : tr(A) \rrbracket(\rho).$$

- Case $\varphi = St$. In this case, $A = N$. Let $n = \llbracket \Gamma \vdash_\mu t : N \rrbracket_\mu(\theta)$. By the induction hypothesis, we have

$$\llbracket tr(\Gamma) \vdash tr(t) : tr(N) \rrbracket(\rho) = \left(\frac{1}{2^n}, 1 - \frac{1}{2^n}, -, - \right).$$

By the definition of $tr(\varphi)$ and calculation, we have

$$\llbracket tr(\Gamma) \vdash tr(\varphi) : tr(A) \rrbracket(\rho) = \left(\frac{1}{2^{n+1}}, 1 - \frac{1}{2^{n+1}}, -, - \right),$$

which implies $\llbracket \Gamma \vdash_\mu St : A \rrbracket_\mu(\theta) = n + 1 \sim_N \llbracket tr(\Gamma) \vdash tr(\varphi) : tr(A) \rrbracket(\rho)$.

- Case $\varphi = (s \leq t)$. In this case, $A = \Omega$. Let $n = \llbracket \Gamma \vdash_\mu s : N \rrbracket_\mu(\theta)$ and $m = \llbracket \Gamma \vdash_\mu t : N \rrbracket_\mu(\theta)$. By the induction hypothesis, we have

$$\llbracket tr(\Gamma) \vdash tr(s) : tr(N) \rrbracket(\rho) = \left(\frac{1}{2^n}, 1 - \frac{1}{2^n}, -, - \right)$$

$$\llbracket tr(\Gamma) \vdash tr(t) : tr(N) \rrbracket(\rho) = \left(\frac{1}{2^m}, 1 - \frac{1}{2^m}, -, - \right).$$

By the definition of $tr(s \leq t)$ and calculation, we have

$$\llbracket tr(\Gamma) \vdash tr(s \leq t) : tr(A) \rrbracket(\rho) = \begin{cases} (1, \rightarrow, \rightarrow, \rightarrow) & \text{(if } \frac{1}{2} \times (\frac{1}{2^n} + (1 - \frac{1}{2^m})) \geq \frac{1}{2}, \text{ i.e. if } n \leq m) \\ (0, \rightarrow, \rightarrow, \rightarrow) & \text{(if } \frac{1}{2} \times (\frac{1}{2^n} + (1 - \frac{1}{2^m})) < \frac{1}{2}, \text{ i.e., if } n > m). \end{cases}$$

Thus, we have $\llbracket \Gamma \vdash_\mu s \leq t : A \rrbracket_\mu(\theta) \sim_A \llbracket tr(\Gamma) \vdash tr(s \leq t) : tr(A) \rrbracket(\rho)$ as required.

- Case $\varphi = \varphi_1 \wedge \varphi_2$.

In this case, we have $A = \Omega$.

By the induction hypothesis, we have

$$\llbracket \Gamma \vdash_\mu \varphi_i : A \rrbracket_\mu(\theta) \sim_A \llbracket tr(\Gamma) \vdash tr(\varphi_i) : tr(A) \rrbracket(\rho)$$

for each $i = 1, 2$.

By the definition of \sim_Ω , we have

$$\llbracket tr(\Gamma) \vdash tr(\varphi_i) : tr(\Omega) \rrbracket(\rho) = (\llbracket \Gamma \vdash_\mu \varphi_i : \Omega \rrbracket_\mu(\theta), \rightarrow, \rightarrow, \rightarrow)$$

for each $i = 1, 2$. Therefore we have

$$\begin{aligned} \llbracket \Gamma \vdash_\mu \varphi_1 \wedge \varphi_2 : \Omega \rrbracket_\mu(\theta) &= \llbracket \Gamma \vdash_\mu \varphi_1 : \Omega \rrbracket_\mu(\theta) \sqcap_\Omega \llbracket \Gamma \vdash_\mu \varphi_2 : \Omega \rrbracket_\mu(\theta) \\ &\sim_\Omega (\llbracket \Gamma \vdash_\mu \varphi_1 : \Omega \rrbracket_\mu(\theta) \sqcap_\Omega \llbracket \Gamma \vdash_\mu \varphi_2 : \Omega \rrbracket_\mu(\theta), \rightarrow, \rightarrow, \rightarrow) \\ &= \llbracket tr(\Gamma) \vdash \varphi_1 : Prop \rrbracket(\rho) \sqcap \llbracket tr(\Gamma) \vdash \varphi_2 : Prop \rrbracket(\rho) \\ &= \llbracket tr(\Gamma) \vdash \varphi_1 \wedge \varphi_2 : Prop \rrbracket(\rho) \end{aligned}$$

as desired.

- Case $\varphi = \varphi_1 \vee \varphi_2$. Similar to the above case.
- Case $\varphi = \lambda X. \varphi'$. In this case, A is of the form $B \rightarrow T$, with $\Gamma, X : B \vdash_\mu \varphi' : T$. For any $v \in \llbracket B \rrbracket$ and $u \in \llbracket tr(B) \rrbracket$ such that $v \sim_B u$, we have:

$$\begin{aligned} \llbracket \Gamma \vdash_\mu \varphi : A \rrbracket_\mu(\theta)(v) &= \llbracket \Gamma, X : B \vdash_\mu \varphi' : T \rrbracket_\mu(\theta[X \mapsto v]) \\ &\sim_T \llbracket tr(\Gamma, X : B) \vdash tr(\varphi') : tr(T) \rrbracket(\rho[X \mapsto u]) \\ &\quad \text{(by the induction hypothesis)} \\ &= \llbracket tr(\Gamma) \vdash tr(\varphi) : tr(A)x \rrbracket(\theta)(u) \end{aligned}$$

Therefore, we have

$$\llbracket \Gamma \vdash_\mu \varphi : B \rightarrow T \rrbracket_\mu(\theta) \sim_{B \rightarrow T} \llbracket tr(\Gamma) \vdash tr(\varphi) : tr(B \rightarrow T) \rrbracket(\rho)$$

as required.

- Case $\varphi = \varphi_1 \varphi_2$. We have $A = T$, with $\Gamma \vdash_\mu \varphi_1 : B \rightarrow T$ and $\Gamma \vdash_\mu \varphi_2 : B$.

By the induction hypothesis, we have $\llbracket \Gamma \vdash_\mu \varphi_1 : B \rightarrow T \rrbracket_\mu(\theta) \sim_{B \rightarrow T} \llbracket tr(\Gamma) \vdash tr(\varphi_1) : tr(B \rightarrow T) \rrbracket(\rho)$ and $\llbracket \Gamma \vdash_\mu \varphi_2 : B \rrbracket_\mu(\theta) \sim_B \llbracket tr(\Gamma) \vdash tr(\varphi_2) : tr(B) \rrbracket(\rho)$. Therefore by the definition of $\sim_{B \rightarrow T}$, we have

$$\begin{aligned} \llbracket \Gamma \vdash_\mu \varphi_1 \varphi_2 : A \rrbracket_\mu(\theta) &= \llbracket \Gamma \vdash_\mu \varphi_1 : B \rightarrow T \rrbracket_\mu(\theta) (\llbracket \Gamma \vdash_\mu \varphi_2 : B \rrbracket_\mu(\theta)) \\ &\sim_A \llbracket tr(\Gamma) \vdash tr(\varphi_1) : tr(B \rightarrow T) \rrbracket(\rho) (\llbracket tr(\Gamma) \vdash tr(\varphi_2) : tr(B) \rrbracket(\rho)) \\ &= \llbracket tr(\Gamma) \vdash tr(\varphi_1 \varphi_2) : tr(A) \rrbracket(\rho) \end{aligned}$$

as required.

- Case $\varphi = \mu X. \varphi'$.

In this case, $A = T$, with $\Gamma, X : T \vdash_\mu \varphi' : T$. By the induction hypothesis, for any $v \in \llbracket T \rrbracket_\mu$ and $u \in \llbracket tr(T) \rrbracket$ such that $v \sim_T u$, we have

$$\llbracket \Gamma, X : T \vdash_\mu \varphi' : T \rrbracket_\mu(\theta[X \mapsto v]) \sim_T \llbracket tr(\Gamma), X : tr(T) \vdash tr(\varphi') : tr(T) \rrbracket(\rho[X \mapsto u]).$$

Since $tr(\mu X.\varphi') = \mu X.tr(\varphi')$, it suffices to show:

$$\llbracket \Gamma \vdash_{\mu} \mu X.\varphi' : T \rrbracket_{\mu}(\theta) \sim_T \llbracket tr(\Gamma) \vdash \mu X.tr(\varphi') : tr(T) \rrbracket(\rho).$$

Let $\mathcal{F} : \llbracket T \rrbracket_{\mu} \rightarrow \llbracket T \rrbracket_{\mu}$ and $\mathcal{G} : \llbracket tr(T) \rrbracket \rightarrow \llbracket tr(T) \rrbracket$ be the functions defined by:

$$\mathcal{F}(v) := \llbracket \Gamma, X : T \vdash_{\mu} \varphi' : T \rrbracket_{\mu}(\theta[X \mapsto v])$$

$$\mathcal{G}(u) := \llbracket tr(\Gamma), X : tr(T) \vdash tr(\varphi') : tr(T) \rrbracket(\rho[X \mapsto u]).$$

By the reasoning above, we have $\mathcal{F} \sim_{T \rightarrow T} \mathcal{G}$. By the definitions of the semantics, we have $\llbracket \Gamma \vdash_{\mu} \mu X.\varphi' : T \rrbracket_{\mu}(\theta) = LFP(\mathcal{F})$ and $\llbracket tr(\Gamma) \vdash \mu X.tr(\varphi') : tr(T) \rrbracket(\rho) = LFP(\mathcal{G})$. Then there exists an ordinal α such that

$$LFP(\mathcal{F}) = \mathcal{F}^{\alpha}(\perp_T) \quad \text{and} \quad LFP(\mathcal{G}) = \mathcal{G}^{\alpha}(\perp_{tr(T)}),$$

where $f^{\beta}(x)$ is defined by $f^0(x) = x$, $f^{\beta+1} = f(f^{\beta}(x))$, and $f^{\beta} = \bigsqcup_{\gamma < \beta} f^{\gamma}(x)$ if β is a limit ordinal. We shall prove by (transfinite) induction on β that $\mathcal{F}^{\beta}(\perp_T) \sim_T \mathcal{G}^{\beta}(\perp_{tr(T)})$, which would imply

$$LFP(\mathcal{F}) = \mathcal{F}^{\alpha}(\perp_T) \sim_T \mathcal{G}^{\alpha}(\perp_{tr(T)}) = LFP(\mathcal{G})$$

as required.

The base case $\mathcal{F}^0(\perp_T) = \perp_T \sim_T \perp_{tr(T)} = \mathcal{G}^0(\perp_{tr(T)})$ follows by a straightforward induction on the structure of T . The case where β is a successor ordinal follows immediately from the induction hypothesis and $\mathcal{F} \sim_{T \rightarrow T} \mathcal{G}$. If β is a limit ordinal, then

$$\mathcal{F}^{\beta}(\perp_T) = \bigsqcup_{\gamma < \beta} \mathcal{F}^{\gamma}(\perp_T) \quad \text{and} \quad \mathcal{G}^{\beta}(\perp_T) = \bigsqcup_{\gamma < \beta} \mathcal{G}^{\gamma}(\perp_T).$$

By the induction hypothesis (of the transfinite induction),

$$\mathcal{F}^{\gamma}(\perp_T) \sim_T \mathcal{G}^{\gamma}(\perp_T)$$

for every $\gamma < \beta$. By Lemma A.1 below, we have

$$\mathcal{F}^{\beta}(\perp_T) \sim_T \mathcal{G}^{\beta}(\perp_T)$$

as required.

- Case $\varphi = \nu X.\varphi'$. Similar to the case for $\varphi = \mu X.\varphi'$ above. □

A.2. Proof of Lemma 5.10. The proof proceeds by induction on the derivation for $\mathcal{K}; \Delta \vdash_M \phi : \kappa$, with case analysis on the last rule used.

- Cases for T-WEAKTU and T-WEAK: Trivial by the induction hypothesis.
- Cases for T-AP and T-VAR: Since $\llbracket \lambda \Delta.\phi \rrbracket(\rho)$ does not depend on ρ , we have $\llbracket \lambda \Delta.\phi \rrbracket(\rho) = \llbracket \lambda \Delta.\phi \rrbracket(\rho')$.
- Case for T-FVAR: In this case, $\Delta = \emptyset$ and $\phi = X$. We have

$$\llbracket \lambda \Delta.\phi \rrbracket(\rho) = \rho(X) \lesssim_{\Delta \rightarrow \kappa} \rho'(X) = \llbracket \lambda \Delta.\phi \rrbracket(\rho'),$$

as required.

- Case for T-MU: In this case, $\Delta = \emptyset$ and $\phi = \mu X.\phi'$, with $\mathcal{K}, X : \kappa; \emptyset \vdash_M \phi' : \kappa$, where κ is of the form $\dots \rightarrow Prop^{T, \emptyset}$. Let $h_0 = \lambda x.\llbracket \phi \rrbracket \rho \{X \mapsto x\}$ and $h_1 = \lambda x.\llbracket \phi' \rrbracket \rho' \{X \mapsto x\}$. By the induction hypothesis, $x_0 \preceq_{\kappa} x_1$ implies $h_0 x_0 \preceq_{\kappa} h_1 x_1$. Thus, by Lemma 5.9, we have

$$\llbracket \lambda \Delta.\phi \rrbracket(\rho) = LFP(h_0) \lesssim_{\kappa} LFP(h_1) = \llbracket \lambda \Delta.\phi \rrbracket(\rho'),$$

as required.

- Case for T-NU: Similar to the case for T-MU above.

- Case for T-ABS: In this case, $\phi = \lambda X.\phi'$. The result follows immediately from the induction hypothesis, as $\lambda\Delta.\lambda X.\phi' = \lambda(\Delta, X : Prop^{T,U}).\phi'$.
- Case for T-APP: In this case, $\phi = \phi_1\phi_2$. By the induction hypothesis, we have: $\llbracket \lambda\Delta.\phi_1 \rrbracket(\rho) \lesssim_{\Delta \rightarrow Prop^{T,U} \rightarrow \kappa} \llbracket \lambda\Delta.\phi_1 \rrbracket(\rho')$ and $\llbracket \lambda\Delta.\phi_2 \rrbracket(\rho) \lesssim_{\Delta \rightarrow Prop^{T,U}} \llbracket \lambda\Delta.\phi_2 \rrbracket(\rho')$. Suppose $\Delta = X_1 : Prop^{T_1, U_1}, \dots, X_k : Prop^{T_k, U_k}$ and let $x_i \in \llbracket Prop^{T_i, U_i} \rrbracket$ for each $i \in \{1, \dots, k\}$. Then, we have:

$$\begin{aligned}
& \llbracket \lambda\Delta.\phi \rrbracket(\rho) \\
&= \lambda x_1 \in \llbracket Prop \rrbracket. \dots \lambda x_k \in \llbracket Prop \rrbracket. \llbracket \phi \rrbracket \rho \{X_1 \mapsto x_1, \dots, X_k \mapsto x_k\} \\
&= \lambda x_1 \in \llbracket Prop \rrbracket. \dots \lambda x_k \in \llbracket Prop \rrbracket. (\llbracket \lambda\Delta.\phi_1 \rrbracket \rho x_1 \dots, x_k) (\llbracket \lambda\Delta.\phi_2 \rrbracket \rho x_1 \dots, x_k) \\
&\lesssim_{\Delta \rightarrow \kappa} \lambda x_1 \in \llbracket Prop \rrbracket. \dots \lambda x_k \in \llbracket Prop \rrbracket. (\llbracket \lambda\Delta.\phi_1 \rrbracket \rho' x_1 \dots, x_k) (\llbracket \lambda\Delta.\phi_2 \rrbracket \rho' x_1 \dots, x_k) \\
&= \llbracket \lambda\Delta.\phi \rrbracket(\rho')
\end{aligned}$$

as required.

- The remaining cases follow immediately from the induction hypothesis.