Hennessy-Milner Results for Probabilistic PDL

Tao Gu

Department of Computer Science
University College London
London, UK

Alexandra Silva

Department of Computer Science
University College London
London, UK

Fabio Zanasi

Department of Computer Science
University College London
London, UK

Abstract

Kozen introduced probabilistic propositional dynamic logic (PPDL) in 1985 as a compositional framework to reason about probabilistic programs. In this paper we study expressiveness for PPDL and provide a series of results analogue to the classical Hennessy-Milner theorem for modal logic. First, we show that PPDL charaterises probabilistic trace equivalence of probabilistic automata (with outputs). Second, we show that PPDL can be mildly extended to yield a characterisation of probabilistic state bisimulation for PPDL models. Third, we provide a different extension of PPDL, this time characterising probabilistic event bisimulation.

Keywords: Probabilistic Propositional Dynamic Logic, Probabilistic bisimulation, Hennessy-Milner property

1 Introduction

Probabilistic programming is an extension of imperative programming that enables the specification and implementation of randomized network and security protocols, machine learning and quantum algorithms. The variety of applications has recently led to a rapidly growing interest in the probabilistic perspective. Reasoning about the correctness of such programs, and more generally verifying properties such as convergence and termination, is quite intricate. It is thus important to establish formal techniques that enable these forms of reasoning.

The origins of the formal semantics of probabilistic programs can be traced back to the early 80's. The seminal work by Kozen [13,14] describes how to use Markov Kernels to give precise denotational semantics to simple imperative probabilistic programs. Probabilistic programs allow to encode conditionals and iterations parametric on a coin flip: e.g., "execute program p with probability 0.3 and program q with probability 0.7". Thus, their semantics is not simply a relation from inputs to outputs, but rather a map from an initial state to a (sub-)distribution over possible final states. Reasoning about correctness of such programs differs from the classical setting in that one moves from a Boolean — is a property true or not — to a quantitative perspective — is a property true with high probability.

Probabilistic propositional dynamic logic (PPDL) [14] establishes a framework for expressing and verifying properties of probabilistic programs, which moves from the traditional truth-functional interpretation to a quantitative one. Many properties of probabilistic programs, like termination, can be encoded and verified

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

in PPDL. The box and diamond modalities in PPDL can be thought of as probabilistic analogues of Dijkstra's weakest preconditions [5]. McIver and Morgan [15], and more recently, also Katoen and Kaminski [12,16], have developed a quantitative analogue of Dijkstra's calculus to reason about termination of probabilistic programs, which is closely related to PPDL. In addition to applications in verification, PPDL has found its way to other areas, e.g. it has been used to reason about uncertainty and knowledge in multi-agent systems [8].

On the other hand, Hennessy and Milner first noticed a relationship between bisimulation of labelled transition systems (LTS) and a simple modal logic, subsequently referred to as Hennessy-Milner logic (HML) [10]. In particular, they proved that HML characterises bisimilarity (the largest bisimulation) within the class of image-finite LTS: two states in an image-finite LTS are bisimilar if and only if they satisfy exactly the same HML formulas. The existence of such characterisation of bisimilarity has practical implications for verification of properties: if two states of a system belong to the same HM class, then they can be checked for bisimulation equivalence by looking at HML formulas instead. Moreover, and perhaps more interestingly, if two states are not bisimilar, then one can find an HML formula to witness the failure of (and serve as counter-example to) bisimilarity. For a simple logic such as HML, this is a considerable advantage.

Since the seminal paper of Hennessy and Milner, analogous characterisations have been studied for other logics and systems. In particular, there has been growing interest in the quantitative setting. For example, an HML-style simple logic, called $\mathcal{L}_0$, was introduced in [3,2] to characterise state and event bisimilarity for labelled Markov processes (LMPs). Later on, Desharnais et al. [4] proposed a real-valued logic which gives the same characterisation result for LMPs. Doberkat [6] studied stochastic Kripke models, and introduced (Boolean-valued) stochastic PDL to characterise behavioural equivalence of such models.

In this work, we continue this line of research by studying Hennessy-Milner properties for PPDL. The main contributions of the paper (and its technical roadmap) are as follows:

(i) First, we show that PPDL formulas (or, more precisely, well-structured programs) characterise probabilistic trace equivalence of PPDL models. These are the probabilistic analogues of Kripke models — probabilistic automata with a continuous state space and multiple output functions (Section 3). Not unexpectedly, the challenge here is to prove that trace equivalence implies PPDL equivalence.

(ii) In Section 4, we show that a small extension of PPDL, which we call PPDL$^+$, characterises probabilistic state bisimilarity for PPDL models (with analytical state spaces). PPDL$^+$ extends PPDL with additional function constructors (for Boolean functions) of the form $(-) > r$, where r is arbitrary non-negative rational.

(iii) In Section 5, we show that a different extension of PPDL, which we call PPDL$^\sharp$, characterises probabilistic event bisimilarity for PPDL models. Note that, differently from state bisimilation, this result does not require the state spaces to be analytical.

2 Preliminaries

In this section, we fix some basic notation and recall the necessary background on Markov Kernels, PPDL, and labelled Markov processes.

2.1 Measurable spaces, Markov Kernels, and Categories.

A *measurable space* is a set X equipped with a σ-algebra Σ_X on X: $\Sigma_X \subseteq \mathcal{P}(X)$ is a collection of subsets of X such that it includes X, is closed under complement, and is closed under countable intersections. When no confusion arises, we shall often refer to a measurable space simply by its underlying set. Given two measurable spaces (X, Σ_X) and (Y, Σ_Y), we define:

- A function $f\colon X \to Y$ is *measurable* if $f^{-1}(B) \in \Sigma_X$ whenever $B \in \Sigma_Y$.
- A function $\mu\colon \Sigma_X \to [0, +\infty]$ is a *measure* if it satisfies the countable additivity property: for any countable family $\{A_i \mid i \in I\}$ of pairwise disjoint subsets of X,

$$\mu\left(\bigcup_{i\in I} A_i\right) = \sum_{i\in I} \mu(A_i)$$

  In particular, a *(sub-)probability measure* is one such that $\mu(X) = 1$ (or $\mu(X) \leq 1$).
- A function $h\colon X \times \Sigma_Y \to [0, 1]$ is a *(sub-)Markov kernel* if:
  (i) $h(\cdot, B)\colon X \to [0, 1]$ is a measurable function for any $B \in \Sigma_Y$.
  (ii) $h(x, \cdot)\colon \Sigma_Y \to [0, 1]$ is a (sub-)probability measure for any $x \in X$.

Finally, we recall the definition of *analytic space* — for more details, see e.g. [7]. A metric space $\mathcal{M}$ is *complete* if every Cauchy sequence of points in $\mathcal{M}$ has a limit in $\mathcal{M}$, and *separable* if $\mathcal{M}$ contains a countable

dense subset. A *Polish space* is the topological space underlying a complete and separable metric space. Suppose (X, Σ_X) is Polish. A subset C of X is analytic in X if it is the continuous image of some Polish space. A measurable space (Y, Σ_Y) is *analytic* if it is measurably isomorphic to some analytic set C in a Polish space (X, Σ_X).

2.2 Probabilistic propositional dynamic logic (PPDL)

We now recall the basic syntax and semantics for PPDL from [14]. A PPDL signature is a pair of finite sets $(\mathbb{P}, \mathbb{F})$, where $\mathbb{P}$ and $\mathbb{F}$ are respectively the sets of primitive programs and primitive functions. The programs, functions and formulas of PPDL are defined as:

$$\begin{array}{rr} \text{Booleans} & B ::= \mathbf{1} \mid F \in \mathbb{F} \mid \neg B \mid B \wedge B \mid B \vee B \\ \text{programs} & p, q ::= P \in \mathbb{P} \mid p; q \mid B? \mid ap + bq \mid p^* \\ \text{functions} & f, g ::= \mathbf{1} \mid F \in \mathbb{F} \mid af + bg \mid B \cdot f \mid \langle p \rangle f \\ \text{formulas} & \phi ::= f \leq g \end{array}$$

where $a, b \in \mathbb{Q}_{\geq 0}$ are non-negative rational numbers. Often we shall omit "$\cdot$" and ";" when the context is clear, so that Bf and pq stand for $B \cdot f$ and $p; q$, respectively.

Remark 2.1 In the paper originally introducing PPDL [14], arbitrary linear combinations of functions are allowed. In our presentation, we admit only positive linear combinations for the sake of simplicity. Note that eliminating such restriction would not affect the results that we prove.

Remark 2.2 As we will see, PPDL formulas do not actually play a role in the results of the next sections, which instead revolve on PPDL functions. We still included formulas in our presentation for two reasons. First, to adhere to the original definition of PPDL, as in [14]. Second, to contrast formulas with the function constructor $(-) > r$ that we will introduce in Section 4, see Remark 4.1 below.

To emphasise how PPDL programs capture standard programming constructs, we shall also use the following abbreviations:

$$\textbf{if } B \textbf{ then } p \textbf{ else } q \coloneqq (B?; p) + (\neg B?; q) \qquad \textbf{while } B \textbf{ do } p \coloneqq (B?; p)^*; (\neg B)?$$

In this paper we will focus on the fragment of *well-structured* programs, where the usage of linear combination and iteration is restricted:

$$\text{programs} \qquad p, q ::= P \mid p; q \mid B? \mid \textbf{if } B \textbf{ then } p \textbf{ else } q \mid \textbf{while } B \textbf{ do } p$$

Such restriction, which already appears in [14], has a natural justification. Intuitively, the Kleene star p^* of a program p describes finite iterations of execution of p, which may not always converge to a finite value for all inputs. The restriction to well-structured programs enforces the semantics of p^* (and all programs as given above) to be defined everywhere and return a real value in $[0, 1]$ (for a full proof see e.g. [17]).Whenever we want to emphasise such restriction, we will refer to the fragment of PPDL whose programs are well-structured as well-structured PPDL.

A PPDL model for the signature $\mathcal{L} = (\mathbb{P}, \mathbb{F})$ is a tuple $\mathcal{X} = (X, \Sigma_X, \mathsf{V}^{\mathbb{P}}, \mathsf{V}^{\mathbb{F}})$, where:

- (X, Σ_X) is a measurable space (called the state space), and X is called the set of states.
- $\mathsf{V}^{\mathbb{P}}$ assigns to every $P \in \mathbb{P}$ a sub-Markov kernel $X \times \Sigma_X \to [0, 1]$.
- $\mathsf{V}^{\mathbb{F}}$ assigns to every $F \in \mathbb{F}$ a measurable function $X \to \{0, 1\}$.

The semantics $[\![\cdot]\!]^{\mathcal{X}}$ for well-structured programs and functions are inductively defined as follows:

- Programs are interpreted as sub-Markov kernels $X \times \Sigma_X \to [0, 1]$:

$$[\![P]\!]^{\mathcal{X}} = \mathsf{V}_{\mathbb{P}}(P) \qquad [\![\textbf{if } B \textbf{ then } p \textbf{ else } q]\!]^{\mathcal{X}} = [\![B?; p]\!]^{\mathcal{X}} + [\![(\neg B)?; q]\!]^{\mathcal{X}}$$

$$[\![p; q]\!]^{\mathcal{X}} = \lambda(x, A). \int_{y \in X} [\![p]\!]^{\mathcal{X}}(x, dy) \cdot [\![q]\!]^{\mathcal{X}}(y, A) \qquad [\![\textbf{while } B \textbf{ do } p]\!]^{\mathcal{X}} = \sum_{i \in \mathbb{N}} [\![(B?; p)^i; (\neg B)?]\!]^{\mathcal{X}}$$

$$[\![B?]\!]^{\mathcal{X}} = \lambda(x, A). [\![B]\!]^{\mathcal{X}}(x) \cdot \chi_A(x)$$

where $(B?; p)^0 = \mathbf{1}?$, $(B?; p)^{i+1} = (B?; p)^i; (B?; p)$, and χ_A is the characteristic function for set A.

- Boolean functions are interpreted as $\{0,1\}$-valued measurable functions:

$$\begin{aligned} [\![F]\!]^{\mathcal{X}} &= \mathsf{V}_{\mathbb{F}}(F) & [\![B_0 \wedge B_1]\!]^{\mathcal{X}} &= \lambda x.[\![B_0]\!]^{\mathcal{X}}(x) \wedge [\![B_1]\!]^{\mathcal{X}}(x) \\ [\![\neg B]\!]^{\mathcal{X}} &= \lambda x.1 - [\![B]\!]^{\mathcal{X}}(x) & [\![B_0 \vee B_1]\!]^{\mathcal{X}} &= \lambda x.[\![B_0]\!]^{\mathcal{X}}(x) \vee [\![B_1]\!]^{\mathcal{X}}(x) \end{aligned}$$

- Functions are interpreted as measurable functions $X \to [0, +\infty)$:

$$\begin{aligned} [\![\mathbf{1}]\!]^{\mathcal{X}} &= \lambda x.1 \text{ (the constant function to 1)} & [\![F]\!]^{\mathcal{X}} &= \mathsf{V}_{\mathbb{F}}(F) \\ [\![af + bg]\!]^{\mathcal{X}} &= \lambda x.a \cdot [\![f]\!]^{\mathcal{X}}(x) + b \cdot [\![g]\!]^{\mathcal{X}}(x) & [\![Bf]\!]^{\mathcal{X}} &= \lambda x.[\![B]\!]^{\mathcal{X}}(x) \cdot [\![f]\!]^{\mathcal{X}}(x) \\ [\![\langle p \rangle f]\!]^{\mathcal{X}} &= \lambda x. \int_{y \in X} [\![f]\!]^{\mathcal{X}}(y) \cdot [\![p]\!]^{\mathcal{X}}(x, dy) \end{aligned}$$

- Formulas are interpreted as $\{0,1\}$-valued measurable functions:

$$[\![\varphi \leq \psi]\!]^{\mathcal{X}}(x) = \begin{cases} 1 & \text{if } [\![\varphi]\!]^{\mathcal{X}}(x) \leq [\![\psi]\!]^{\mathcal{X}}(x) \\ 0 & \text{otherwise} \end{cases}$$

A pointed PPDL model is a model $\mathcal{X}$ together with a state $x \in X$. We say two pointed models $(\mathcal{X}, x)$ and $(\mathcal{Y}, y)$ are *equivalent with respect to* PPDL *functions* (denoted as $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}} (\mathcal{Y}, y)$) if $[\![f]\!]^{\mathcal{X}}(x) = [\![f]\!]^{\mathcal{Y}}(y)$, for all functions f. We will also encounter other logics, whose syntax does not have functions but just formulas: with slight abuse of notation, for such logic $\mathcal{L}$, we will use $\equiv_{\mathcal{L}}$ to mean the binary relation that two (pointed) $\mathcal{L}$-models are equivalent with respect to all $\mathcal{L}$-formulas.

2.3 Labelled Markov processes and probabilistic bisimulation.

Given a finite set of actions $\mathcal{A}$, a *labelled Markov process* (LMP) is an $\mathcal{A}$-labelled tuple $(X, \Sigma_X, \tau_a)_{a \in \mathcal{A}}$, where (X, Σ_X) is a measurable space, and for each $a \in \mathcal{A}$, $\tau_a \colon X \times \Sigma_X \to [0,1]$ is a sub-Markov kernel.

We are going to study two different notions of bisimulation for LMPs. Suppose $(X, \Sigma_X, \tau_a)_{a \in \mathcal{A}}$ is an LMP. First, we introduce state bisimulation [18]. For a relation $\mathcal{R} \subseteq X \times X$ and a subset $A \subseteq X$, we say that A is $\mathcal{R}$*-closed* if for any $x \mathcal{R} y$, one has $x \in A$ if and only if $y \in A$. The relation $\mathcal{R} \subseteq X \times X$ is a (LMP) *state bisimulation* if for any $x \mathcal{R} x'$, $a \in \mathcal{A}$, and $\mathcal{R}$-closed $A \in \Sigma_X$,

$$\tau_a(x, A) = \tau_a(x', A).$$

We say two pointed LMPs $(\mathcal{X}, x)$ and $(\mathcal{X}, x')$ are *state bisimilar*, denoted as $(\mathcal{X}, x) \leftrightarrows^s_0 (\mathcal{X}, x')$, if there exists a state bisimulation $\mathcal{R} \subseteq X \times X$ such that $x \mathcal{R} x'$.

There is another notion of bisimulation on LMPs, called event bisimulation [2]. A sub-σ-algebra $\Lambda \subseteq \Sigma_X$ is an *event bisimulation* if $(X, \Lambda, \tau_a)_{a \in \mathcal{A}}$ is also a LMP. Such Λ generates a binary relation $\mathfrak{R}(\Lambda)$, such that $(x, x') \in \mathfrak{R}(\Lambda)$ if $x \in A$ precisely when $x' \in A$, for all $A \in \Lambda$. With a bit abuse of terminology, we may also refer to this $\mathfrak{R}(\Lambda)$ as an event bisimulation when Λ is. We say two pointed LMPs $(\mathcal{X}, x)$ and $(\mathcal{X}, x')$ are *event bisimilar*, denoted as $(\mathcal{X}, x) \leftrightarrows^e_0 (\mathcal{X}, x')$, if there exists an event bisimulation Λ such that $(x, x') \in \mathfrak{R}(\Lambda)$.

It is worth noting that state and event bisimulation coincide in discrete settings and for analytic state spaces. However, in general, the two notions are orthogonal [2,20]. From a categorical point of view, LMP state bisimulation are spans in the category of LMPs with analytic state spaces (seen as coalgebras). Note that the spans are properly definable (as a weak pullback) only when the LMPs are analytic [3,18]. On the other hand, LMP event bisimulations are cospans of surjections in the category of LMPs, which are definable without any need of the analyticity restriction [2].

2.4 Logical characterisation of bisimilarity for LMPs.

For our later developments, it is important to recall two Hennessy-Milner results in the context of LMPs. The simple logic $\mathcal{L}_0$ [3] is defined inductively as follows:

$$\phi ::= \top \mid \phi \wedge \phi \mid \langle a \rangle_r \phi \tag{1}$$

where $a \in \mathcal{A}$ is an action and r is a rational number in $[0,1)$. A model for $\mathcal{L}_0$ is a LMP $(X, \Sigma_X, \tau_a)_{a \in \mathcal{A}}$. Let $[\![\cdot]\!]^{\mathcal{X}}_{\mathcal{L}_0}$ denote the interpretation of $\mathcal{L}_0$ in $\mathcal{X}$, or simply $[\![\cdot]\!]$ if the context is clear. Then $[\![\top]\!] := X$, $[\![\phi \wedge \psi]\!] := [\![\phi]\!] \cap [\![\psi]\!]$, and $[\![\langle a \rangle_r \phi]\!] := \{x \in X \mid \tau_a(x, [\![\phi]\!]) > r\}$.

There are two important results (which we will use in Section 4 and 5): (1) if we consider LMPs whose state spaces are analytic, then $\mathcal{L}_0$ characterises LMP state bisimilarity [18]. (2) if we consider LMPs in general, then $\mathcal{L}_0$ characterises LMP event bisimilarity [2].

Proposition 2.3 ([18]) *For a countable action set $\mathcal{A}$, suppose $\mathcal{X} = (X, \Sigma_X, \tau_a)_{a\in\mathcal{A}}$ is an LMP with an analytic state space, and x, y are two states in X. Then $(\mathcal{X}, x) \leftrightarroweq^s_0 (\mathcal{X}, y)$ if and only if $(\mathcal{X}, x) \equiv_{\mathcal{L}_0} (\mathcal{X}, y)$.*

Proposition 2.4 ([2]) *Suppose that $\mathcal{X} = (X, \Sigma_X, \tau_a)_{a\in\mathcal{A}}$ is an LMP, and x, y are two states in X. Then $(\mathcal{X}, x) \leftrightarroweq^e_0 (\mathcal{X}, y)$ if and only if $(\mathcal{X}, x) \equiv_{\mathcal{L}_0} (\mathcal{X}, y)$.*

3 PPDL characterises trace equivalence

In this section we provide a Hennessy-Milner property for PPDL (functions). First, we introduce the notion of trace equivalence between PPDL models, which is a generalization of that for probabilistic automata (PA). Then, we prove that PPDL characterises trace equivalence. Note that trace equivalence is a relatively weak notion, so the non-trivial direction is that trace equivalence implies PPDL equivalence (Prop. 3.5).

The first question to tackle is how to define a notion of trace equivalence for PPDL models in a principled way. To this aim, we use the observation that a PPDL model can be seen as a (continuous) probabilistic automaton (PA) with multiple output functions. The standard trace semantics of a PA is the set of its trace distributions, each of which is a probability distribution assigning to a certain set of traces a probability value [19]. Two pointed PAs are trace equivalent if the two states have exactly the same trace distributions. Motivated by this, we define trace distributions and trace equivalence between pointed PPDL models. Since we now work in the continuous case, the trace semantics consists of sets of probability measures (while in the discrete case we reason with sets of probability distributions).

To present our definition, we fix a finite signature $(\mathbb{P}, \mathbb{F})$. We define the alphabet as $\Sigma := \mathbb{P} \uplus \mathcal{B}?$, where $\mathcal{B}$ is the set of all Boolean combinations of $F \in \mathbb{F}$, and $\mathcal{B}? = \{B? \mid B \in \mathcal{B}\}$; we use Σ^* to denote the set of all finite words over Σ; in particular, ϵ is the empty word.

Remark 3.1 Note that, since $\mathbb{F}$ is finite, there are only finitely many elements in $\mathcal{B}$ modulo logic equivalence $\equiv$. With slight abuse of notation, henceforth we use $\mathcal{B}?$ to denote a finite set of letters, containing one fixed $B?$ from each $\equiv$-equivalence class in $\mathcal{B}$. The choice of the representative of equivalence classes does not affect our results. Hence the alphabet Σ is finite.

Definition 3.2 [Trace] A PPDL model $\mathcal{X} = (X, \Sigma_X, \mathsf{V}_{\mathbb{P}}, \mathsf{V}_{\mathbb{F}})$ determines, for each word $\omega \in \Sigma^*$, a sub-Markov kernel $\tau_\omega \colon X \times \Sigma_X \to [0, 1]$, inductively as follows. For arbitrary $x \in X$ and $A \in \Sigma_X$:

$$\begin{aligned}
\tau_\epsilon(x, A) &:= \chi_A(x) \\
\tau_P(x, A) &:= \mathsf{V}_{\mathbb{P}}(P)(x, A) \text{ where } P \in \mathbb{P} \\
\tau_{B?}(x, A) &:= \begin{cases} 1 & [\![B]\!]^{\mathcal{X}}(x) = 1 \text{ and } x \in A \\ 0 & \text{otherwise, namely } [\![B]\!]^{\mathcal{X}}(x) = 0 \text{ or } x \notin A \end{cases} \\
\tau_{\omega\cdot a}(x, A) &:= \int_{y\in X} \tau_\omega(x, dy) \cdot \tau_a(y, A), \text{ where } \omega \in \Sigma^*, \text{ and } a \in \Sigma
\end{aligned}$$

Then the *trace at state x* is a function $\rho_x \colon \Sigma^* \times \mathbb{F} \to \mathbb{R}$ such that given $\omega \in \Sigma^*$ and $F \in \mathbb{F}$,

$$\rho_x(\omega, F) := \int_{y\in X} \tau_\omega(x, dy) \cdot \mathsf{V}_{\mathbb{F}}(F)(y)$$

Given two pointed PPDL models $(\mathcal{X}, x)$ and $(\mathcal{Y}, y)$, we say they are *trace equivalent* (denoted as $(\mathcal{X}, x) \approx_{tr} (\mathcal{Y}, y)$) if $\rho^{\mathcal{X}}_x = \rho^{\mathcal{Y}}_y$, namely if $\rho^{\mathcal{X}}_x$ and $\rho^{\mathcal{Y}}_y$ coincide on all inputs $(\omega, F) \in \Sigma^* \times \mathbb{F}$.

Towards a characterisation result, our first observation is that PPDL equivalence entails trace equivalence.

Proposition 3.3 $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}} (\mathcal{Y}, y)$ *implies* $(\mathcal{X}, x) \approx_{tr} (\mathcal{Y}, y)$.

Proof. By Definition 3.2 we have $(\mathcal{X}, x) \approx_{tr} (\mathcal{Y}, y)$ if and only if that $\rho_x(\omega, F) = \rho_y(\omega, F)$ holds for arbitrary primitive function $F \in \mathbb{F}$ and finite word $\omega \in \Sigma^*$. Note that every value $\rho_x(\omega, F)$ is exactly the interpretation of a PPDL formula $\langle\omega\rangle F$ at the state x, so $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}} (\mathcal{Y}, y)$ implies $\rho_x(\omega, F) = \rho_y(\omega, F)$, for all ω and F. □

For the converse direction, namely trace equivalence implies PPDL equivalence, we need the following factorisation lemma.

Lemma 3.4 (Factorisation) *Every* PPDL *function f is equivalent to some* PPDL *function g (in the sense that $[\![f]\!]^{\mathcal{X}} = [\![g]\!]^{\mathcal{X}}$ for all models $\mathcal{X}$) of the following form:*

$$\begin{aligned} G &::= \mathbf{1} \mid F \mid B \mid \langle p \rangle G \\ g &::= G \mid ag + bg \mid B \cdot g \end{aligned} \tag{2}$$

where p is a program, $a, b \in [0, +\infty)$. In other words, one can always push the appearance of programs in PPDL *functions to "the innermost layer".*

Proof. The proof goes by induction on the complexity of PPDL functions. We show the only non-trivial case, which is $f = \langle p \rangle f'$ for some f'. First, let us write $\simeq$ for semantic equivalence between PPDL functions, namely $f \simeq g$ if $[\![f]\!]^{\mathcal{X}} = [\![g]\!]^{\mathcal{X}}$ for all models $\mathcal{X}$. We then prove by cases of the structure of f'. If $f' = \mathbf{1}$ or $f' = F$ for a primitive F, then $f = \langle p \rangle f'$ is already in the correct shape. If $f' = af_0 + bf_1$, we have $f = \langle p \rangle (af_0 + bf_1) \simeq a\langle p \rangle f_0 + b\langle p \rangle f_1$, and we can apply the induction hypothesis on f_0 and f_1. If $f' = B \cdot f''$, note that $f = \langle p \rangle B \cdot f'' \simeq \langle p \rangle \langle B? \rangle f'' \simeq \langle p; B? \rangle f''$, and again we can conclude using the induction hypothesis on f''. Finally, if $f' = \langle p' \rangle f''$, we have that $f = \langle p \rangle \langle p' \rangle f'' \simeq \langle p; p' \rangle f''$, and we can conclude by induction hypothesis on f''. □

We are now ready to show that trace equivalence implies PPDL equivalence.

Proposition 3.5 $(\mathcal{X}, x) \approx_{tr} (\mathcal{Y}, y)$ *implies* $(\mathcal{X}, x) \equiv_{\text{PPDL}} (\mathcal{Y}, y)$.

Proof. Suppose $(\mathcal{X}, x) \approx_{tr} (\mathcal{Y}, y)$. It suffices to prove that $(\mathcal{X}, x)$ and $(\mathcal{Y}, y)$ agree on all functions g in Lemma 3.4. We reason by induction on the structure of g. We first consider the case where g is some G, defined as in (2). This has 4 sub-cases:

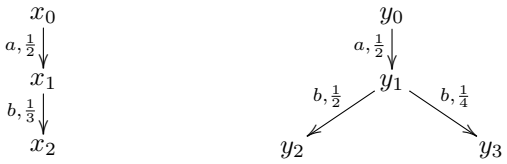
- $G = \mathbf{1}$: trivial.
- G is a primitive function F: since $(\mathcal{X}, x) \approx_{tr} (\mathcal{Y}, y)$, they agree on the empty word ϵ. In particular, $\rho_x(\epsilon, F) = \rho_y(\epsilon, F)$.
- G is some Boolean combination B of primitive functions. This case follows from the fact that the value of B is totally determined by its component primitive functions.
- $G = \langle p \rangle F$, where $F \in \mathbb{F}$. We first show that every program p can be written as a (countable) sum of words in Σ^*. Then $[\![\langle p \rangle F]\!]^{\mathcal{X}}(x) = [\![\langle p \rangle F]\!]^{\mathcal{Y}}(y)$ is reduced to $[\![\langle \omega \rangle F]\!]^{\mathcal{X}}(x) = [\![\langle \omega \rangle F]\!]^{\mathcal{Y}}(y)$ for any $\omega \in \Sigma^*$, which is exactly the definition of trace equivalence. We reason by induction on the structure of the programs.
  - The cases for $p = P$ and $p = B?$ are trivial, as they are already in the alphabet.
  - As for the cases for $p =$ **if** B **then** p_0 **else** p_1 and $p =$ **while** B **do** q, one simply spells out the definitions.
  - $p = p_0; p_1$. By induction hypothesis, both p_0 and p_1 can be written as sums of words in Σ^*, say $p_0 = \sum_{i \in I} \omega_i$, $p_1 = \sum_{j \in J} \pi_j$, then $p_0; p_1 = \sum_{i \in I, j \in J} \omega_i; \pi_j$.

Next we consider the remaining two cases for g, namely $g = ag_0 + bg_1$ and $g = B \cdot g'$. Both can be proved by straightforward application of the induction hypothesis. □

Propositions 3.5 and 3.3 together yield the desired Hennessy-Milner property for PPDL:

Theorem 3.6 (PPDL characterises trace equivalence) *For arbitrary pointed* PPDL *models $(\mathcal{X}, x)$ and $(\mathcal{X}, y)$, $(\mathcal{X}, x) \equiv_{\text{PPDL}} (\mathcal{X}, y)$ if and only if $(\mathcal{X}, x) \approx_{tr} (\mathcal{X}, y)$.*

Example 3.7 Let the signature be $\mathbb{P} = \{a, b\}$ and $\mathbb{F} = \{F_1, F_2\}$. Consider the following two pointed PPDL models $(\mathcal{X}, x_0)$ and $(\mathcal{Y}, y_0)$:



Also, define $F_1(x_2) = F_1(y_2) = F_1(y_3) = 1$, $F_2(x_2) = F_2(y_2) = 1$, and for all the other unmentioned cases F_1 and F_2 have value 0. Then $(\mathcal{X}, x_0)$ and $(\mathcal{Y}, y_0)$ are not trace equivalent. Theorem 3.6 tells us that there exists some PPDL function f that has different values on x_0 and y_0. For example, let $f = \langle a \rangle \langle b \rangle (F_1 \wedge F_2)$, then

$$[\![f]\!]^{\mathcal{X}}(x_0) = \frac{1}{2} \times \frac{1}{3} \times 1 = \frac{1}{6} \qquad [\![f]\!]^{\mathcal{Y}}(y_0) = \frac{1}{2} \times \frac{1}{2} \times 1 + \frac{1}{2} \times \frac{1}{2} \times 0 = \frac{1}{4}$$

4 Extended PPDL characterises state bisimilarity

Theorem 3.6 characterises trace equivalence in terms of PPDL. It is a natural question to ask what extra logical structure is needed in order to characterise state bisimularity and event bisimilarity for PPDL models. In this section we focus on state bisimulation (and leave event bisimulation to the next section). To this aim, we introduce a suitable extension of PPDL, called PPDL$^+$, and we show that it characterises state bisimilarity over PPDL models with analytic state spaces.

4.1 PPDL$^+$

We start with introducing the logic PPDL$^+$ as an extension of PPDL. The idea is that, since $\mathcal{L}_0$ characterises LMP state bisimilarity (Proposition 2.3), we extend PPDL so that it can interpret $\mathcal{L}_0$ formulas as functions. In the grammar below, we point out which clauses are the same as in PPDL to emphasise the difference between the two logics. PPDL$^+$ is defined as

$$\begin{array}{rr} \text{PPDL}^+ \text{ Booleans} & B ::= \mathbf{1} \mid F \in \mathbb{F} \mid \neg B \mid B \wedge B \mid B \vee B \mid f > r \\ \text{PPDL}^+ \text{ programs} & p, q ::= P \in \mathbb{P} \mid p; p \mid B? \mid ap + bq \mid p^* \\ \text{PPDL}^+ \text{ functions} & f, g ::= B \mid af + bg \mid B \cdot f \mid \langle p \rangle f \\ \text{PPDL}^+ \text{ formulas} & \phi ::= f \leq g \end{array}$$

where $a, b \in [0, +\infty)$, and r is arbitrary rational number in $[0, +\infty)$. As in the case of PPDL, we always restrict to the well-structured programs for PPDL$^+$. The resulting logic PPDL$^+$ has the same set of programs as PPDL, but with additional function constructors (for Boolean functions) of the form $(-) > r$.

Remark 4.1 Note that this new function constructor apparently resembles the shape of the constructor of PPDL formulas, but it is of a different nature: $(-) > r$ is fixed for each rational number $r \in \mathbb{Q}_{\geq 0}$, and it yields a *function*, whereas $(-) \leq (-)$ in the PPDL syntax acts on two variable arguments, and it yields a *formula*.

The logic PPDL$^+$ is also interpreted on PPDL models. The (boolean) function $f > r$ is interpreted as a $\{0, 1\}$-valued function such that $[\![f > r]\!]^{\mathcal{X}}(x) = 1$ if $[\![f]\!]^{\mathcal{X}}(x) > r$, and 0 otherwise (namely if $[\![f]\!]^{\mathcal{X}}(x) \leq r$).

4.2 State bisimulation

We now develop the characterisation result for state bisimilarity. For the reminder of this subsection, we restrict ourselves to those LMPs (and PPDL models) whose state spaces are analytic. The notion of state bisimulation for PPDL models is defined as the extension of the same notion on the underlying LMPs, by taking the extra weight structure into account.

Definition 4.2 [State bisimulation for PPDL models] Let $\mathcal{X}$ be a PPDL model. A *state bisimulation* on $\mathcal{X}$ is an equivalence relation $\mathcal{R} \subseteq X \times X$, such that if $x \mathcal{R} x'$, then

(i) $\mathsf{V}_{\mathbb{P}}(P)(x, A) = \mathsf{V}_{\mathbb{P}}(P)(x', A)$, for any $P \in \mathbb{P}$ and $\mathcal{R}$-closed $A \in \Sigma_X$ (for the definition of $\mathcal{R}$-closure, see Section 2).

(ii) $\mathsf{V}_{\mathbb{F}}(F)(x) = \mathsf{V}_{\mathbb{F}}(F)(x')$, for any $F \in \mathbb{F}$.

Given two states $x, x' \in X$, we say the two pointed PPDL models $(\mathcal{X}, x)$ and $(\mathcal{X}, x')$ are *state bisimilar* (denoted as $(\mathcal{X}, x) \leftrightarrows^s (\mathcal{X}, x')$) if there exists a state bisimulation $\mathcal{R} \subseteq X \times X$ such that $x \mathcal{R} x'$.

Remark 4.3 Recall that, when defining trace equivalence (Definition 3.2), we let the primitive functions (and their boolean combinations) be the labels of the LMPs. This allows us to account for tests of the form $B?$, where B is some boolean combination of primitive functions. Only with this setup the resulting notion of trace equivalence implies logical equivalence. For bisimulation, such extra label structure is unnecessary, whence the above definition.

Although the above definition only concerns a single model $\mathcal{X}$, for bisimulation between two models $\mathcal{X}$ and $\mathcal{Y}$, one can simply apply the definition to their disjoint union $\mathcal{X} \uplus \mathcal{Y}$. It is an immediate observation that condition (i) above, can be extended to τ_ω for arbitrary program $\omega \in \Sigma^*$ (where $\Sigma = \mathbb{P} \uplus \mathcal{B}?$ as defined in Section 3):

Lemma 4.4 *If $(\mathcal{X}, x) \leftrightarrows^s (\mathcal{X}, x')$ by $\mathcal{R}$, then $\tau_\omega(x, A) = \tau_\omega(x', A)$, for any $\omega \in \Sigma^*$ and $\mathcal{R}$-closed $A \in \Sigma_X$.*

Proof. We prove the statement by induction on ω.

- $\omega = \epsilon$: we need to show that $x \in A$ if and only if $x' \in A$, which holds because $x \mathcal{R} x'$ and set A is $\mathcal{R}$-closed.

- $\omega = P$, where $P \in \mathbb{P}$: we need to show that $\mathsf{V}_{\mathbb{P}}(P)(x, A) = \mathsf{V}_{\mathbb{P}}(P)(x', A)$, and this is already (part of) the definition of state bisimulation.
- $\omega = B?$, where $B \in \mathcal{B}$: the desired equation is reduced to $\mathsf{V}_{\mathbb{F}}(B)(x) = \mathsf{V}_{\mathbb{F}}(B)(x')$, and $x \in A$ if and only if $x' \in A$. They follow immediately from condition 2 in Definition 4.2 and that A is $\mathcal{R}$-closed.
- $\omega = \omega' \cdot a$, where $\omega' \in \Sigma^*$ and $a \in \Sigma$: there are two sub-cases.
  (i) $a = P \in \mathbb{P}$. First, recall that $\tau_{\omega' \cdot P}(x, A) = \int_{y \in X} \tau_{\omega'}(x, dy) \cdot \mathsf{V}_{\mathbb{P}}(P)(y, A)$. Note that $(\mathsf{V}_{\mathbb{F}}(P)(-, A))^{-1}(r)$ is always $\mathcal{R}$-closed, for arbitrary $r \in [0, 1]$. So by induction hypothesis,

$$\tau_{\omega'}(x, \mathsf{V}_{\mathbb{F}}(a)(-, A)^{-1}(r)) = \tau_{\omega'}(x', \mathsf{V}_{\mathbb{F}}(a)(-, A)^{-1}(r))$$

  which implies that the two integrals are equivalent.
  (ii) $a = B?$, where $B \in \mathcal{B}$. Similarly, note that $\tau_{\omega' \cdot B?}(x, A) = \int_{y \in X} \tau_{\omega'}(x, dy) \cdot \tau_{B?}(y, A)$, and that $(\tau_{B?}(-, A))^{-1}(r)$ is always $\mathcal{R}$-closed, for arbitrary $r \in [0, 1]$.

□

The next observation is that, as in the non-probabilistic setting, state bisimulation implies trace equivalence.

Lemma 4.5 $(\mathcal{X}, x) \leftrightarroweq^s (\mathcal{X}, x')$ *implies* $(\mathcal{X}, x) \approx_{tr} (\mathcal{X}, x')$.

Proof. Suppose $\mathcal{R} \subseteq X \times X$ is a state bisimulation on $\mathcal{X}$ such that $x \mathcal{R} x'$. We make a case distinction on $\omega \in \Sigma^*$ to show that $\rho_x(\omega, F) = \rho_{x'}(\omega, F)$, for any $F \in \mathbb{F}$.

- $\omega = \epsilon$: $\rho_x(\epsilon, F) = \mathsf{V}_{\mathbb{F}}(F)(x)$, and by Definition 4.2 $\mathsf{V}_{\mathbb{F}}(F)(x) = \mathsf{V}_{\mathbb{F}}(F)(x')$.
- ω is non-empty: $\rho_x(\omega, F) = \int_{y \in X} \tau_\omega(x, dy) \mathsf{V}_{\mathbb{F}}(F)(y) = \int_{y \in \mathsf{V}_{\mathbb{F}}(F)^{-1}(1)} \tau_\omega(x, dy) = \tau_\omega(x, \mathsf{V}_{\mathbb{F}}(F)^{-1}(1))$. Note that $\mathsf{V}_{\mathbb{F}}(F)^{-1}(1)$ is obviously $\mathcal{R}$-closed, so by Lemma 4.4 we have $\tau_\omega(x, \mathsf{V}_{\mathbb{F}}(F)^{-1}(1)) = \tau_\omega(x', \mathsf{V}_{\mathbb{F}}(F)^{-1}(1))$.

□

The above observation, paired with our characterisation of trace equivalence in terms of PPDL (Theorem 3.6), yields the question of how PPDL fails to characterise the stronger notion of state bisimilarity, thus making necessary the introduction of PPDL$^+$. The following counterexample illustrates this point.

Example 4.6 Consider the PPDL signature $(\mathbb{P} = \{a, b\}, \mathbb{F} = \{F\})$, and the following two pointed models $(\mathcal{X}, x_0)$ and $(\mathcal{Y}, y_0)$:

where $F(x_1) = F(x_2) = F(x_3) = F(y_1) = F(y_3) = 1$, $F(x_4) = F(y_4) = 0$, and $F(x_0) = F(y_0) = 0$. We claim that $x_0 \equiv_{\mathsf{PPDL}} y_0$. By Theorem 3.6, it suffices to check that $(\mathcal{X}, x_0)$ and $(\mathcal{Y}, y_0)$ are trace equivalent. For example, $\rho_{x_0}(a, F) = \frac{1}{2} \times 1 + \frac{1}{2} \times 1 = 1$, and $\rho_{y_0}(a, F) = 1 \times 1 = 1$. However, the two pointed models are not state bisimilar. This is because if such $\mathcal{R}$ exists, then $\mathcal{R}$ should also contain (x_1, y_1) and (x_2, y_1), but this does not work because y_1 has a non-trivial probabilistic branching, which neither x_1 nor x_2 has.

For the non-trivial direction of the characterisation, namely $\equiv_{\mathsf{PPDL}^+}$ implies state bisimilarity, the idea is to start from some simple logic $\mathcal{L}$, resembling $\mathcal{L}_0$ (see Section 2), such that $\equiv_{\mathcal{L}}$ implies state bisimilarity. If we can show that PPDL$^+$ can express $\mathcal{L}$, then $\equiv_{\mathsf{PPDL}^+}$ implies $\equiv_{\mathcal{L}}$, and we are done. This motivates us to introduce $\mathcal{L}_1$ as an extension of $\mathcal{L}_0$, based on the observation that the main difference between PPDL models and LMPs is that the former has some extra weight structure (namely $\mathsf{V}^{\mathbb{F}}$). Differently from $\mathcal{L}_0$, in $\mathcal{L}_1$ we have those primitive functions F in $\mathbb{F}$ as primitive formulas:

$$\mathcal{L}_1 \ni \phi ::= \top \mid F \mid \phi \wedge \phi \mid \langle P \rangle_r \phi \tag{3}$$

where $F \in \mathbb{F}$, $P \in \mathbb{P}$, and r is arbitrary rational number in $[0, 1)$. The logic $\mathcal{L}_1$ is interpreted on PPDL models, and the semantics is similar to that for $\mathcal{L}_0$ on LMP. In particular $(\mathcal{X}, x) \models F$ if $\mathsf{V}_{\mathbb{F}}(F)(x) = 1$. We can show the following proposition as the analogue of Proposition 2.3 for the case of PPDL model. The proof can be found in Appendix A.

Proposition 4.7 *Suppose that $\mathcal{X} = (X, \Sigma_X, \mathsf{V}^{\mathbb{P}}, \mathsf{V}^{\mathbb{F}})$ is a* PPDL *model with an analytic space, and x, y are two states in X. Then $(\mathcal{X}, x) \leftrightarroweq^s (\mathcal{X}, y)$ if and only if $(\mathcal{X}, x) \equiv_{\mathcal{L}_1} (\mathcal{X}, y)$.*

Then, in order to show that PPDL$^+$ equivalence imply state bisimilarity, it suffices to show that PPDL$^+$ can encode $\mathcal{L}_1$ (as defined in (3)). In the following lemma, let $\approx$ denote the binary relation between $\mathcal{L}_1$ formulas and PPDL$^+$ (boolean) functions, such that $\phi \approx f$ if and only if they are semantically equivalent: for any PPDL model $\mathcal{X}$ and $x \in X$, $\mathcal{X}, x \vDash \phi$ if and only if $[\![f]\!]^{\mathcal{X}}(x) = 1$, and $\mathcal{X}, x \nvDash \phi$ if and only if $[\![f]\!]^{\mathcal{X}}(x) = 0$.

Lemma 4.8 *For every $\mathcal{L}_1$ formula ϕ, there exists some* PPDL$^+$ *Boolean function f such that $\phi \approx f$.*

Proof. We reason by induction on $\mathcal{L}_1$ formulas ϕ. Note that $\top \approx \mathbf{1}$; $F \approx F$; $\phi_1 \wedge \phi_2 \approx f_1 \wedge f_2$, where $\phi_i \approx f_i$, $i = 1, 2$. For the case $\phi = \langle a \rangle_r \psi$, suppose by induction hypothesis that $\psi \approx f$. Then one may check that $\langle a \rangle_r \psi \approx (\langle a \rangle f) > r$. □

We are now ready to prove the main result of this subsection.

Theorem 4.9 (PPDL$^+$ characterises state bisimilarity) *For arbitrary pointed* PPDL *model $\mathcal{X}$ with analytic state spaces and states $x, y \in X$, $(\mathcal{X}, x) \leftrightarroweq^s (\mathcal{X}, y)$ if and only if $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}^+} (\mathcal{X}, y)$.*

Proof. For the forward direction, we reason by induction on functions f. Given Lemma 4.5 and Proposition 3.5 , it only remains to show the case in which $f = f' \geq r$. Suppose $(\mathcal{X}, x) \leftrightarroweq^s (\mathcal{X}, y)$. By inductive hypothesis, $[\![f']\!](x) = [\![f']\!](y)$, so $[\![f']\!](x) \geq r$ if and only if $[\![f']\!](y) \geq r$. This entails $[\![f > r]\!](x) = [\![f > r]\!](y)$.

For the other direction, suppose $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}^+} (\mathcal{X}, y)$. We construct a binary relation $\mathcal{R} \subseteq X \times X$, and check the two conditions in Definition 4.2. By Lemma 4.8, $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}^+} (\mathcal{X}, y)$ implies $(\mathcal{X}, x) \equiv_{\mathcal{L}_1} (\mathcal{X}, y)$. So simply let $\mathcal{R}$ be $\equiv_{\mathcal{L}_1}$, and $x \mathcal{R} y$. Then apply Proposition 4.7, we know that $(\mathcal{X}, x) \leftrightarroweq^s (\mathcal{X}, y)$ under bisimulation $\mathcal{R}$. □

Example 4.10 We recall Example 4.6, where $(\mathcal{X}, x_0)$ are and $(\mathcal{X}, y_0)$ are not state bisimilar. Then Theorem 4.9 implies that there is some PPDL$^+$ function that can distinguish the two pointed models. For example, let $f = \langle a \rangle \left(\langle b \rangle (F) > \frac{1}{3} \right) > \frac{2}{3}$. Then one can calculate that:

$$\left[\!\!\left[\langle b \rangle F > \frac{2}{3} \right]\!\!\right]^{\mathcal{X}} (x_1) = 1 \qquad \left[\!\!\left[\langle b \rangle F > \frac{2}{3} \right]\!\!\right]^{\mathcal{X}} (x_2) = \left[\!\!\left[\langle b \rangle F > \frac{2}{3} \right]\!\!\right]^{\mathcal{Y}} (y_1) = 0$$

which entails that $[\![f]\!]^{\mathcal{X}}(x_0) = 1$ and $[\![f]\!]^{\mathcal{Y}}(y_0) = 1$. So f can distinguish $(\mathcal{X}, x_0)$ and $(\mathcal{Y}, y_0)$.

5 Extended PPDL characterises event bisimilarity

This section is devoted to showing that another mild extension of PPDL can be used to characterise *event* bisimilarity for PPDL models. The notion of event bisimulation was proposed in [2] as a more appropriate way than state bisimulation to define behavioural equivalence for LMPs, because it does not require the underlying state spaces to be analytic. In what follows, we first introduce the new logic, then the notion of event bisimulation for PPDL models, and finally show their correspondence.

5.1 PPDL$^\sharp$

The logic characterising event bisimilarity for PPDL models will be called PPDL$^\sharp$. Intuitively, PPDL$^\sharp$ is obtained by adding $\mathcal{L}_0$ to PPDL in a "minimal" way: unlike PPDL$^+$, where $(-) > r$ can be applied to arbitrary functions, in PPDL$^\sharp$ only certain instances of $f > r$ are admitted. More precisely, PPDL$^\sharp$ programs p, p' and PPDL$^\sharp$ functions f are defined as follows, where we highlight when the grammar has the same clauses as PPDL.

$$\begin{aligned}
&\text{PPDL Booleans} && B ::= \mathbf{1} \mid F \in \mathbb{F} \mid B \wedge B \mid B \vee B \mid \neg B && (4)\\
&\text{PPDL}^\sharp\text{ Booleans} && C ::= B \mid \langle P \rangle (C) > r \mid C \wedge C && (5)\\
&\text{PPDL}^\sharp\text{ programs} && p ::= P \in \mathbb{P} \mid p; p \mid B? \mid aP + bP \mid p^* &\\
&\text{PPDL functions} && g, g' ::= B \mid ag + bg' \mid B \cdot g \mid \langle p \rangle g && (6)\\
&\text{PPDL}^\sharp\text{functions} && f ::= g \mid C && (7)\\
&\text{PPDL formulas} && \phi ::= f \leq g &
\end{aligned}$$

As in the case of PPDL, we always restrict to the well-structured programs. Note a PPDL$^\sharp$ function f is defined to be either a PPDL function g, or a function C in which a new function constructor $\langle P \rangle (\cdot) > r$ may appear.

Intuitively, the reason for considering PPDL$^\sharp$ instead of PPDL$^+$ is that event bisimulation is too weak to preserve arbitrary functions of type $\langle p\rangle f$, as allowed by PPDL$^+$ syntax, and instead requires functions to enjoy some structural property, as the one guaranteed by Lemma 3.4 for PPDL functions.

The semantics of PPDL$^\sharp$ is based on PPDL models: all the programs and functions are interpreted as in PPDL, whereas the new function construct $\langle P\rangle(C) > r$ is interpreted as a measurable $\{0,1\}$-valued function

$$[\![\langle P\rangle(C) > r]\!](x) = \begin{cases} 1 & \text{if } [\![P]\!](x, \{u \in X \mid [\![C]\!](u) = 1\}) > r \\ 0 & \text{otherwise} \end{cases}.$$

5.2 Event bisimulation

As for state bisimulation, we can define event bisimulation for PPDL models as a mild extension on the same notion on LMPs.

Definition 5.1 [Event bisimulation for PPDL models] An *event bisimulation* on a PPDL model $\mathcal{X} = (X, \Sigma_X, \mathsf{V}^{\mathbb{P}}, \mathsf{V}^{\mathbb{F}})$ is a sub-σ-algebra Λ of Σ_X such that $(X, \Lambda, \mathsf{V}^{\mathbb{P}}, \mathsf{V}^{\mathbb{F}})$ is also a PPDL model.

In other words, Λ is an event bisimulation on $\mathcal{X}$ if Λ is an event bisimulation on the LMP $(X, \Sigma_X, \mathsf{V}^{\mathbb{P}}(P))_{P\in\mathbb{P}}$, and $\mathsf{V}^{\mathbb{F}}(F) \in \Lambda$ for all $F \in \mathbb{F}$. Note that Λ generates an equivalence relation $\mathfrak{R}(\Lambda)$ on X: $x\mathfrak{R}(\Lambda)y$ if and only if $x \in A$ precisely when $y \in A$, for all $A \in \Lambda$. We say two states x and y are *event bisimilar* (denoted as $(\mathcal{X}, x) \leftrightarroweq^e (\mathcal{X}, y)$) if there exists an event bisimulation Λ such that $x\mathfrak{R}(\Lambda)y$. Intuitively, two states x and y are event bisimilar by Λ if they cannot be separated by any set in Λ.

Remark 5.2 Just as for state bisimulation (Lemma 4.5), one may prove that event bisimulation entails trace equivalence. We do not elaborate further on this result, as it is not needed for the sequel.

We now move on to proving the characterisation result for event bisimilarity. The idea is similar to that for state bisimilarity, but here the non-trivial direction is that PPDL$^\sharp$ is invariant under event bisimulation. We present a simple logic $\mathcal{L}_1^{\neg,\vee}$ which is invariant under (actually, characterises) event bisimulation. Then we show that, $\mathcal{L}_1^{\neg,\vee}$ is strong enough to "express" PPDL$^\sharp$ (Lemma 5.7).

The (Boolean) logic $\mathcal{L}_1^{\neg,\vee}$ is defined by adding Boolean operators $\neg$ and $\vee$ to $\mathcal{L}_1$:

$$\mathcal{L}_1^{\neg,\vee} \ni \phi ::= \top \mid F \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \langle P\rangle_r\phi \tag{8}$$

where $F \in \mathbb{F}$, $P \in \mathbb{P}$, and r ranges over all rational numbers in $[0,1)$. The semantics of $\mathcal{L}_1^{\neg,\vee}$ on PPDL models is the same as that of $\mathcal{L}_1$ w.r.t. $\top$, F's, $\wedge$ and modality; $\vee$ and $\neg$ are interpreted as boolean operaters as usual.

We now establish a characterisation result of the logic $\mathcal{L}_1^{\neg,\vee}$. First, we need to introduce some terminology. With a bit abuse of notation, we use $[\![\cdot]\!]_{\mathcal{L}_1^{\neg,\vee}}$ to denote the interpretation of $\mathcal{L}_1^{\neg,\vee}$, or simply $[\![\cdot]\!]$ when the context is clear. We also need the notion of *stability* from [2]: a set $\Lambda \subseteq \Sigma$ is stable w.r.t. Σ if for all $P \in \mathbb{P}$, $A \in \Lambda$, and all rational numbers $r \in [0,1]$, the set $\langle P\rangle_r(A) := \{y \in X \mid \mathsf{V}^{\mathbb{P}}(P)(y, A) > r\}$ is in Λ. We use $[\![\mathcal{L}_1]\!]$ to denote the set $\{[\![\phi]\!] \mid \phi \in \mathcal{L}_1\} \subseteq \mathcal{P}(X)$ of interpretations of all $\mathcal{L}_1$ formulas, and likewise for $\mathcal{L}_1^{\neg,\vee}$. Recall that a π-system $\mathcal{C}$ on set X is a subset of $\mathcal{P}(X)$ such that $X \in \mathcal{C}$ and $\mathcal{C}$ is closed under intersection. Given $\mathcal{C} \subseteq \mathcal{P}(X)$, we use $\sigma(\mathcal{C})$ to denote the σ-algebra generated by $\mathcal{C}$. We would like to point out that the following proof of Proposition 5.5 via Lemma 5.3 and 5.4 has a similar structure to the proof of that $\mathcal{L}_0$ characterises LMP event bisimulation in [2].

Lemma 5.3 *$[\![\mathcal{L}_1]\!]$ is the smallest stable π-system w.r.t. Σ_X that contains $[\![F]\!]$ for all $F \in \mathbb{F}$.*

Proof. To see that $[\![\mathcal{L}_1]\!]$ is a π-system, simply note that $X = [\![\top]\!]$ is in $[\![\mathcal{L}_1]\!]$, and $\wedge$ is interpreted as intersection. To see that $[\![\mathcal{L}_1]\!]$ is stable, given $[\![\phi]\!] \in [\![\mathcal{L}_1]\!]$, then $\langle P\rangle_r([\![\phi]\!]) = [\![\langle P\rangle_r\phi]\!] \in [\![\mathcal{L}_1]\!]$. It is the smallest because if $\mathcal{C}$ is a stable π-system containing all $[\![F]\!]$,

- $[\![\top]\!] = X \in \mathcal{C}$.
- $[\![F]\!] \in \mathcal{C}$, for all $F \in \mathbb{F}$.
- If $[\![\phi]\!], [\![\psi]\!] \in \mathcal{C}$, then $[\![\phi \wedge \psi]\!] = [\![\phi]\!] \cap [\![\psi]\!] \in \mathcal{C}$, since $\mathcal{C}$ is closed under intersection.
- If $[\![\phi]\!] \in \mathcal{C}$, $P \in \mathbb{F}$, and rational number $r \in [0,1)$, then $[\![\langle P\rangle_r\phi]\!] = \langle P\rangle_r([\![\phi]\!]) \in \mathcal{C}$, since $\mathcal{C}$ is stable.

□

Lemma 5.4 *$\sigma([\![\mathcal{L}_1]\!])$ is the smallest stable σ-algebra included in Σ_X that contains all $[\![F]\!]$s.*

Proof. First we know that if $\mathcal{C}$ is a stable π-system, then $\sigma(\mathcal{C})$ is also stable (Lemma 5.4, [2]). So $\sigma([\![\mathcal{L}_1]\!])$ is a stable σ-algebra included in Σ_X. Let Λ be an arbitrary stable σ-algebra $\Lambda \subseteq \Sigma_X$ containing all $[\![F]\!]$s. Since Λ is also a π-system, by Lemma 5.3 we know that $[\![\mathcal{L}_1]\!] \subseteq \Lambda$. Then $\sigma([\![\mathcal{L}_1]\!]) \subseteq \Lambda$ as well. Since $\sigma([\![\mathcal{L}_1]\!])$ is contained by every such σ-algebra, it is the smallest one. □

Proposition 5.5 ($\mathcal{L}_1^{\neg,\vee}$ characterises event bisimilarity) *Let $\mathcal{X}$ be a* PPDL *model, and x, y are two states in X. Then $(\mathcal{X}, x) \equiv_{\mathcal{L}_1^{\neg,\vee}} (\mathcal{X}, y)$ if and only if $(\mathcal{X}, x) \leftrightarroweq^e (\mathcal{X}, y)$.*

Proof. The first observation is that, since $\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!]) = \sigma([\![\mathcal{L}_1]\!])$, Lemma 5.4 entails that $\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!])$ is the smallest stable σ-algebra included in Σ that contains all $[\![F]\!]$s.

We now prove the claim. On one direction, $\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!])$ is an event bisimulation because it is a stable σ-algebra. This means that if $(\mathcal{X}, x) \equiv_{\mathcal{L}_1} (\mathcal{X}, y)$, then x and y are event bisimilar under $\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!])$.

Conversely, suppose there is an event bisimulation $\Lambda \subseteq \Sigma$ such that $x\mathfrak{R}(\Lambda)y$. Since $\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!])$ is the smallest among all event bisimulations over $\mathcal{X}$, we have $\sigma([\![\mathcal{L}_1]\!]) \subseteq \Lambda$, and $\mathfrak{R}(\Lambda) \subseteq \mathfrak{R}(\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!]))$. Therefore $x\mathfrak{R}(\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!]))y$ as well, which implies that x and y agree on all $\mathcal{L}_1^{\neg,\vee}$ formulas. □

Remark 5.6 It is worth noticing that a similar argument as the one above may show that $\mathcal{L}_1$ (defined as in (3)) also characterises event bisimilarity. Then, one may wonder why we introduced $\mathcal{L}_1^{\neg,\vee}$ in first place. This has to do with our overall aim, of proving that PPDL$^\sharp$ characterises event bisimilarity. The non-trivial direction of this result is that event bisimulation perserves PPDL$^\sharp$ functions. For this purpose, we want to use a logic $\mathcal{L}$ such that event bisimulation preserves $\mathcal{L}$ functions, and $\mathcal{L}$ can "almost" express PPDL$^\sharp$ (for example, a PPDL$^\sharp$ formula can be expressed by an infinite conjunction of $\mathcal{L}$ formulas). It turns out $\mathcal{L}_1$ is too simple for the task, whence the introduction of $\mathcal{L}_1^{\neg,\vee}$.

The next lemma is needed to show that event bisimulation implies PPDL$^\sharp$ equivalence. It focusses on PPDL$^\sharp$ functions of the form $\langle p\rangle\hat{B}$, where $\hat{B}$ is any Boolean (*c.f.* (4)) and $p \in \Sigma^*$. Intuitively, it states that $\mathcal{L}_1^{\neg,\vee}$ can express such functions, provided that we allow infinite conjunctions.

Lemma 5.7 *For all* PPDL$^\sharp$ *formula of the form $\langle p\rangle\hat{B}$ (where $p \in \Sigma^*$), there exist a (possibly infinite) set of $\mathcal{L}_1$ formulas $\Phi(p)$, such that for any Boolean B, any states x, y, if x and y agree on (every formula in) $\Phi(p)[B/\hat{B}]$, then $[\![\langle p\rangle B]\!](x) = [\![\langle p\rangle B]\!](y)$.*

The proof for Lemma 5.7 can be found in Appendix A. We are now ready to prove our Hennessy-Milner result for PPDL$^\sharp$ and event bisimilarity.

Theorem 5.8 (PPDL$^\sharp$ characterises event bisimilarity) *For arbitrary* PPDL *model $\mathcal{X}$ and states $x, y \in X$, $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}^\sharp} (\mathcal{X}, y)$ if and only if $(\mathcal{X}, x) \leftrightarroweq^e (\mathcal{X}, y)$.*

Proof. We first prove the simpler direction. Suppose that $(\mathcal{X}, x) \equiv_{\mathsf{PPDL}^\sharp} (\mathcal{X}, y)$. Note that PPDL$^+$ can encode $\mathcal{L}_1^{\neg,\vee}$, so this implies $(\mathcal{X}, x) \equiv_{\mathcal{L}_1^{\neg,\vee}} (\mathcal{X}, y)$. By Lemma 5.5, x and y are bisimular, as witnessed in particular by the bisimulation $\mathfrak{R}(\sigma([\![\mathcal{L}_1^{\neg,\vee}]\!]))$.

As for the non-trivial direction, suppose $(\mathcal{X}, x) \leftrightarroweq^e (\mathcal{X}, y)$. We need to prove that $[\![f]\!]^{\mathcal{X}}(x) = [\![f]\!]^{\mathcal{Y}}(y)$, for all PPDL$^\sharp$ functions f. Recall that, by definition as in Equation (7), PPDL$^\sharp$ functions can be of two kinds.

- First, f might be a PPDL function (as in (6)). So Lemma 3.4 applies, and we may assume $f = \langle p\rangle B$, where B is some PPDL Boolean. By the semantics of PPDL, it suffices to show that x and y have the same value for all functions of the type $\langle p\rangle B$, where $p \in \Sigma^* = (\mathbb{P} \cup \mathcal{B}?)^*$. This is proved in Lemma 5.7.
- The second case is when f is C, for C defined as in (5). Note that each such C can be expressed as a $\mathcal{L}_1^{\neg,\vee}$ formula, so by Proposition 5.5, $[\![C]\!]^{\mathcal{X}}(x) = [\![C]\!]^{\mathcal{Y}}(y)$ for all Cs.

Therefore for any PPDL$^\sharp$ function f we have $[\![f]\!]^{\mathcal{X}}(x) = [\![f]\!]^{\mathcal{Y}}(y)$. □

Remark 5.9 Note that in the discrete case, the notions of state bisimulation and event bisimulation coincide [2]. So the two pointed models $(\mathcal{X}, x_0)$ and $(\mathcal{Y}, y_0)$ in Example 4.6 are not event bisimilar. According to Theorem 5.8, there is some PPDL$^\sharp$ function that distinguishes the two models. In fact, the function $f = \langle a\rangle\left(\langle b\rangle(F) > \frac{2}{3}\right) > \frac{1}{2}$ from Example 4.10 still works here, since such f is also a PPDL$^\sharp$ function.

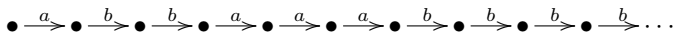
6 Discussion

In this paper we provided three Hennessy-Milner style characterisation results. First, we showed that PPDL functions characterises trace equivalence. Second, we extended PPDL with a new function constructor $(-) > r$

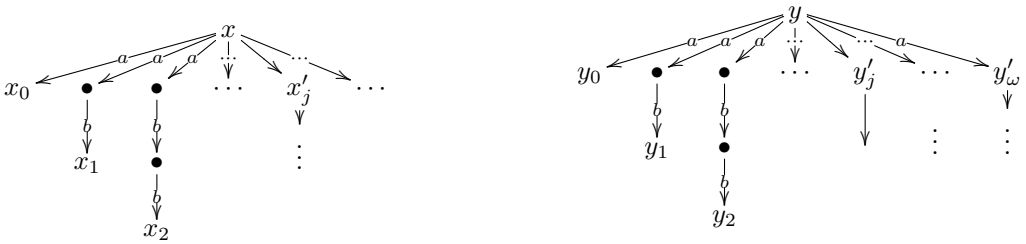
for any rational $r \in [0,1]$, which allows us to express "threshold check" of function values. The resulting logic, PPDL$^+$, characterises state bisimilarity of PPDL models with analytic state spaces. Third, we studied a different extension PPDL$^\sharp$ of PPDL, in which the use of the function constructor $(-) > r$ is restricted: one can only apply multiple constructors of the form $\langle a \rangle (-) > r$ to PPDL Booleans. We proved that PPDL$^\sharp$ characterises event bisimulation between PPDL models. Note that, in the characterisations of PPDL$^+$ and PPDL$^\sharp$, a key step was to show that they can encode the simple logic $\mathcal{L}_0$: this allowed us to exploit the results in in [18], relating $\mathcal{L}_0$ to bisimilarity for LMPs. With respect to [4], we characterised a richer class of models (namely PPDL models instead of LMPs) using logics with extra structure in the modality. Also, we considered both trace bisimilarity and trace equivalence.

Comparison with PDL. In the non-probabilistic case, finiteness of branching of the models plays a role in the characterisation results: PDL, similarly to modal logic [1], characterises only bisimilarity of finitely branching Kripke models but not bisimilarity of arbitrarily branching models. In contrast, such restrictions are not needed in the probabilistic case: in fact, though we chose to present our results using continuous distributions, the same results also hold if one restricts to the discrete case. From a categorical point of view, it means that our results cover models that are elements of the Kleisli category of both the Giry monad $\mathcal{G}_{\leq 1}$ (continuous) and the sub-distribution monad $\mathcal{D}_{\leq 1}$ (discrete). Note that in the non-probabilistic setting, finitely-branching Kripke models are precisely the Kleisli arrows of the finite powerset monad $\mathcal{P}_f$ and possibly infinite-branching Kripke models are the Kleisli arrows of the unrestricted powerset monad $\mathcal{P}$.

To illustrate this, we now provide a counterexample showing that PDL-equivalence does not imply $\mathcal{P}$-bisimulation — to the best of our knowledge this observation is novel. Let the alphabet set be $\Sigma = \{a, b\}$, and fix an infinite string $\omega \in \Sigma^\infty$ as follows:



More precisely, there are $n+1$ appearances of label a right after n appearances of label b, and $n+2$ appearances of label b right after $n+1$ appearances of label a, for all $n \in \mathbb{N}$. Then consider the following two pointed relational models $(\mathcal{X}, x)$ and $(\mathcal{Y}, y)$:



In words, in $(\mathcal{X}, x)$ all the finite paths $x \cdots x_n$ ($n \in \mathbb{N}$) exhaust all the finite initial segments of ω, and all infinite paths $x x'_j \cdots$ exhaust the infinite paths in $\Sigma^\infty \setminus \{\omega\}$. The same holds for $(\mathcal{Y}, y)$, except that y has an extra successor y'_ω, such that the infinite path $y y'_\omega \cdots$ forms the sequence ω. It is not hard to check that $(\mathcal{X}, x)$ and $(\mathcal{Y}, y)$ are equivalent with regard to all PDL formulas, but they are not bisimilar.

Outlook: coalgebraic charaterisations. Finally, we also regard our work as a starting point of further investigation on Hennessy-Milner results for the generic coalgebraic dynamic logic (CoDL) [11,9]. The analogy between the case of CoDL and that of PPDL is summarised by the following table.

	CoDL	PPDL
Category	**Sets**	**Meas**
Monad	$\mathcal{T}$	$\mathcal{G}_{\leq 1}$
$[\![\text{Programs}]\!]$	$X \to \mathcal{T}X$	$X \to \mathcal{G}_{\leq 1}X$
$[\![\text{Propositions/Functions}]\!]$	2^X	$[0, +\infty)^X$

More details on the items appearing in the column for CoDL can be found in [11]. The column for PPDL simply reformulates categorically the presentation of PPDL semantics in Section 2, where **Meas** is the category of measurable spaces and $\mathcal{G}_{\leq 1}$ is the sub-Giry monad on **Meas**.

There are various interesting observations on CoDL that can be made exploiting this analogy. For example, if in a CoDL-system $\mathcal{L}$ (defined in [11]) the proposition operators distribute over modalities, then a similar argument as the one of Proposition 3.5 shows that $\mathcal{L}$ characterises trace equivalence. We leave further developments as future work.

References

[1] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[2] Vincent Danos, Josée Desharnais, Francois Laviolette, and Prakash Panangaden. Bisimulation and cocongruence for probabilistic systems. *Information and Computation*, 204:503–523, 04 2006.

[3] Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. *Inf. Comput.*, 179(2):163–193, 2002.

[4] Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled markov processes. *Theoretical Computer Science*, 318(3):323 – 354, 2004.

[5] Edsger Wybe Dijkstra. *A Discipline of Programming*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1976.

[6] Ernst Doberkat. A stochastic interpretation of propositional dynamic logic: Expressivity. volume 77, pages 50–64, 01 2011.

[7] R. M. Dudley. *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2002.

[8] Ronald Fagin and Joseph Y. Halpern. Reasoning about knowledge and probability. In *Proceedings of the 2Nd Conference on Theoretical Aspects of Reasoning About Knowledge*, TARK '88, pages 277–293, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[9] Helle Hvid Hansen, Clemens Kupke, and Raul Andres Leal. Strong completeness for iteration-free coalgebraic dynamic logics. In *IFIP International Conference on Theoretical Computer Science*, pages 281–295. Springer, 2014.

[10] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[11] Helle Hvid Hansen and Clemens Kupke. Weak completeness of coalgebraic dynamic logics. *Electronic Proceedings in Theoretical Computer Science*, 191, 09 2015.

[12] Nils Jansen, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Federico Olmedo, Friedrich Gretz, and Annabelle McIver. Conditioning in probabilistic programming. *Electr. Notes Theor. Comput. Sci.*, 319:199–216, 2015.

[13] Dexter Kozen. Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, 22(3):328–350, 1981.

[14] Dexter Kozen. A probabilistic pdl. *Journal of Computer and System Sciences*, 30(2):162 – 178, 1985.

[15] Annabelle McIver and Carroll Morgan. Developing and reasoning about probabilistic programs in *pGCL*. In Ana Cavalcanti, Augusto Sampaio, and Jim Woodcock, editors, *Refinement Techniques in Software Engineering, First Pernambuco Summer School on Software Engineering, PSSE 2004, Recife, Brazil, November 23-December 5, 2004, Revised Lectures*, volume 3167 of *Lecture Notes in Computer Science*, pages 123–155. Springer, 2004.

[16] Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. A new proof rule for almost-sure termination. *PACMPL*, 2(POPL):33:1–33:28, 2018.

[17] Prakash Panangaden. Probabilistic relations. *School of Computer Science Research Reports-University of Birmingham CSR*, pages 59–74, 1998.

[18] Prakash Panangaden. *Labelled Markov Processes*. Imperial College Press, London, UK, UK, 2009.

[19] Roberto Segala. A compositional trace-based semantics for probabilistic automata. In Insup Lee and Scott A. Smolka, editors, *CONCUR '95: Concurrency Theory*, pages 234–248, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[20] Pedro Sanchez Terraf. Unprovability of the logical characterization of bisimulation. *Information and Computation*, 209(7):1048 – 1056, 2011.

A Missing Proofs

Proof. [Lemma 4.7] The forward direction is easy, so we do not elaborate it here. As for the other direction, we apply Proposition 2.4. We first construct a LMP $\mathcal{S}$ from $\mathcal{X}$. $\mathcal{S}$ has the action set $\mathcal{A} := \mathbb{P} \uplus \mathbb{F}?$, where $\mathbb{F}? := \{F? \mid F \in \mathbb{F}\}$. The state space (S, Σ_S) of $\mathcal{S}$ is the same as (X, Σ_X). For the transition functions, define:

$$\rho_P(s, A) := \mathsf{V}^{\mathbb{P}}(P)(s, A)$$
$$\rho_{F?}(s, A) := \begin{cases} 1 & \text{if } \mathsf{V}^{\mathbb{F}}(F)(s) = 1 \text{ and } s \in A \\ 0 & \text{otherwise} \end{cases}$$

Then $(X, \Sigma_X, \tau_a)_{a \in \mathcal{A}}$ is a LMP. In the rest of this proof let $\mathcal{L}_0(\mathcal{A})$ be the simple logic $\mathcal{L}_0$ based on this new signature $\mathcal{A}$. Then $(S, \Sigma_S, \tau_a)_{a \in \mathcal{A}}$ is a model for $\mathcal{L}_0(\mathcal{A})$. Suppose $(\mathcal{X}, x) \equiv_{\mathcal{L}_1} (\mathcal{X}, y)$. We claim that $(\mathcal{S}, x) \equiv_{\mathcal{L}_0(\mathcal{A})} (\mathcal{S}, y)$. This is proved by a stronger result: for every $\mathcal{L}_0(\mathcal{A})$ formula ϕ, there exists a $\mathcal{L}_1$ formula $\kappa(\phi)$ such that $(\mathcal{S}, x) \vDash \phi$ if and only if $(\mathcal{X}, x) \vDash \kappa(\phi)$. We reason by induction on the structure of $\mathcal{L}_0(\mathcal{A})$ formulas.

- For $\phi = \top$, we simply let $\kappa(\top) = \top$.
- For $\phi = \phi_1 \wedge \phi_2$, we let $\kappa(\phi_1 \wedge \phi_2) = \kappa(\phi_1) \wedge \kappa(\phi_1)$.
- For $\mathcal{L}_0(\mathcal{A})$ formula ϕ of the form $\langle P\rangle_r\psi$, we let $\kappa(\langle P\rangle_r\psi)$ be $\langle P\rangle_r\kappa(\psi)$.
- For $\mathcal{L}_0(\mathcal{A})$ formula ϕ of the form $\langle F?\rangle_r\psi$, we let $\kappa(\langle F?\rangle_r\psi)$ be $F \wedge \kappa(\psi)$.

We need to check that such κ indeed works. For example, $(\mathcal{S}, x) \vDash \langle F?\rangle_r\psi$ is defined as $\rho_{F?}(x, \{u \in S \mid (\mathcal{S}, u) \vDash \psi\}) > r$. Note that rational number r takes value in $[0, 1)$, so the inequality holds if and only if $\mathsf{V}^{\mathbb{F}}(F)(x) = 1$ and $(\mathcal{S}, x) \vDash \psi$. Then by induction hypothesis we know that this is equivalent to that $(\mathcal{X}, x) \vDash F \wedge \kappa(\psi)$.

Now apply Proposition 2.3 to $(\mathcal{S}, x) \equiv_{\mathcal{L}_0(\mathcal{A})} (\mathcal{S}, y)$, and we know that $(\mathcal{S}, x) \leftrightarrows^s_0 (\mathcal{S}, y)$, where the state bisimulation here is between pointed LMPs. We claim that if $(\mathcal{S}, x) \leftrightarrows^s_0 (\mathcal{S}, y)$ under some LMP state bisimulation $\mathcal{R}$, then $\mathcal{R}$ is also a state bisimulation such that $(\mathcal{X}, x) \leftrightarrows^s (\mathcal{X}, y)$. We simply check the two conditions:

- Let $P \in \mathbb{P}$, and $A \in \Sigma_X$ be $\mathcal{R}$-closed. Then by definition of LMP state bisimulation, we know that $\mathsf{V}^{\mathbb{P}}(P)(x, A) = \rho_P(x, A) = \rho_P(y, A) = \mathsf{V}^{\mathbb{P}}(P)(y, A)$.
- Let $F \in \mathbb{F}$ be a primitive function, and x', y' be two states in $\mathcal{X}$ such that $x'\mathcal{R}y'$. Note that $\mathsf{V}^{\mathbb{F}}(F)(x') = 1$ if and only if $(\mathcal{S}, x') \vDash \langle F?\rangle_{\frac{1}{2}}\top$. Also, $\mathcal{R}$ is a LMP state bisimulation over $\mathcal{S}$ implies that $(\mathcal{S}, x') \equiv_{\mathcal{L}_0(\mathcal{A})} (\mathcal{S}, y')$. In particular, $(\mathcal{X}, x') \vDash \langle F?\rangle_{\frac{1}{2}}\top$ if and only if $(\mathcal{X}, y') \vDash \langle F?\rangle_{\frac{1}{2}}\top$. Therefore $\mathsf{V}^{\mathbb{F}}(F)(x') = \mathsf{V}^{\mathbb{F}}(F)(y')$.

In particular, according to the proof of Proposition 2.3 in [18], we know that $\equiv_{\mathcal{L}_1}$ is a state bisimulation on $\mathcal{X}$. □

Proof. [Lemma 5.7] We reason by induction on the structure of p.

(i) p is empty string. Then $\Phi(\hat{B}) := \{\hat{B}\}$. This is because $[\![B]\!](x) = [\![B]\!](y)$ if and only if $\mathcal{X}, x \vDash B \iff \mathcal{X}, y \vDash B$.

(ii) p is some primitive program $P \in \mathbb{P}$. Then define

$$\Phi(P) := \{\langle P\rangle_r\hat{B} \mid r \text{ is a rational number in } [0, 1)\}$$

To see that this $\Phi(P)$ works, suppose that x and y agree on all formulas in $\Phi(P)[B/\hat{B}]$, but $[\![\langle P\rangle B]\!](x) \neq [\![\langle P\rangle B]\!](y)$. Without loss of generality, we assume that $[\![\langle P\rangle B]\!](x) < [\![\langle P\rangle B]\!](y)$. Then there exists some rational number $r \in [0, 1)$ such that $[\![\langle P\rangle B]\!](x) < r < [\![\langle P\rangle B]\!](y)$. But this means that $\mathcal{X}, x \nvDash \langle P\rangle_r B$ while $\mathcal{X}, y \vDash \langle P\rangle_r B$, contradicting the assumption that x and y agree on all formulas in $\Phi(P)[B/\hat{B}]$.

(iii) p is some test $B?$. Then $\Phi(B?) := \{B \wedge \hat{B}\}$. For arbitrary Boolean B', note that $[\![\langle B?\rangle B']\!]$ is $\{0, 1\}$-valued:

$$\begin{aligned}[\![\langle B?\rangle B']\!](x) = 1 &\iff [\![B]\!](x) = [\![B']\!](x) = 1\\ &\iff \mathcal{X}, x \vDash B \wedge B'.\end{aligned}$$

Therefore, $[\![\langle B?\rangle B']\!](x) = [\![\langle B?\rangle B']\!](y)$ if and only if x and y agree on the formula $B \wedge B'$.

(iv) $p = p'; B?$. Let $\Phi(p) := \{\psi[(B \wedge \hat{B})/\hat{B}] \mid \psi \in \Phi(p')\}$. Let B' be an arbitrary PPDL$^\sharp$ Boolean, and states x and y agree on all formulas in $\Phi(p)[B'/\hat{B}]$. Then we know that for every $\psi \in \Phi(p')$, $x \vDash \psi[(B \wedge B')/\hat{B}]$ if and only if $y \vDash \psi[(B \wedge B')/\hat{B}]$. By induction hypothesis, this means that $[\![\langle p'\rangle(B \wedge B')]\!](x) = [\![\langle p'\rangle(B \wedge B')]\!](y)$, so $[\![\langle p'; B?\rangle B']\!](x) = [\![\langle p'; B?\rangle B']\!](y)$.

(v) $p = p'; Q$, where $Q \in \mathbb{P}$. Let $\Phi(p)$ be the following set:

$$\{\psi[\langle Q\rangle_r\hat{B}/\hat{B}] \mid \psi \in \Phi(p'), r \text{ is rational number in } [0, 1)\}.$$

Let B be an arbitrary PPDL$^\sharp$ Boolean, and x and y be two states that agree on all formulas in $\Phi(p)[B/\hat{B}]$. We add a new pseudo function symbol $\underline{\langle Q\rangle_r B}$, whose value at any state x is the same as the truth value of the $\mathcal{L}_1$ formula $\langle Q\rangle_r$ at state x. Then x and y also agree on $\Phi(p)[\underline{\langle Q\rangle_r B}/\hat{B}]$, for arbitrary rational

$r \in [0,1)$. By induction hypothesis, $\langle p' \rangle \underline{\langle Q \rangle_r B}(x) = \langle p' \rangle \underline{\langle Q \rangle_r B}(y)$, for any rational $r \in [0,1)$. Note that

$$\begin{aligned}\langle p' \rangle \underline{\langle Q \rangle_r B}(x) &= \int_{u \in X} [\![p']\!](x, du) [\![\underline{\langle Q \rangle_r B}]\!](u) \\ &= [\![p']\!](x, [\![\langle Q \rangle_r B]\!])\end{aligned}$$

so we have $[\![p']\!](x, [\![\langle Q \rangle_r B]\!]) = [\![p']\!](y, [\![\langle Q \rangle_r B]\!])$, for all rational numbers $r \in [0,1)$. Lebesgue's convergence theorem tells us the following:

$$\begin{aligned}\langle p'; Q \rangle B(x) &= \int_{u \in X} [\![p']\!](x, du) [\![Q]\!](u, [\![B]\!]) \\ &= \lim_{n \to \infty} \sum_{i=0}^{n-1} [\![p']\!](x, \{u \in X \mid [\![Q]\!](u, [\![B]\!]) \in (\frac{i}{n}, \frac{i+1}{n}]\}) \cdot \frac{i}{n}\end{aligned}$$

in which every set $\{u \in X \mid [\![Q]\!](u, [\![B]\!]) \in (\frac{i}{n}, \frac{i+1}{n}]\}$ is exactly $[\![\langle Q \rangle_{\frac{i+1}{n}} B]\!] \setminus [\![\langle Q \rangle_{\frac{i}{n}} B]\!]$. So

$$\begin{aligned}&\langle p'; Q \rangle B(x) \\ &= \lim_{n \to \infty} \sum_{i=0}^{n-1} [\![p']\!](x, [\![\langle Q \rangle_{\frac{i+1}{n}} B]\!] \setminus [\![\langle Q \rangle_{\frac{i}{n}} B]\!]) \\ &= \lim_{n \to \infty} \sum_{i=0}^{n-1} ([\![p']\!](x, [\![\langle Q \rangle_{\frac{i+1}{n}} B]\!]) - [\![p']\!](x, [\![\langle Q \rangle_{\frac{i}{n}} B]\!])) \\ &= \lim_{n \to \infty} \sum_{i=0}^{n-1} ([\![p']\!](y, [\![\langle Q \rangle_{\frac{i+1}{n}} B]\!]) - [\![p']\!](y, [\![\langle Q \rangle_{\frac{i}{n}} B]\!])) \\ &= \langle p'; Q \rangle B(y)\end{aligned}$$

□