Behavioral Equivalences for Higher-Order Languages with Probabilities

Valeria Vignudelli

Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
Ciclo XXIX

Settore Concorsuale di afferenza: 01/B1
Settore Scientifico Disciplinare: INF/01

Behavioral Equivalences for Higher-Order Languages with Probabilities

Presentata da: Valeria Vignudelli

Coordinatore Dottorato	Relatore
Paolo Ciaccia	Davide Sangiorgi

Esame finale anno 2017

Abstract

Higher-order languages, whose paradigmatic example is the λ-calculus, are languages with powerful operators that are capable of manipulating and exchanging programs themselves. This thesis studies behavioral equivalences for programs with higher-order and probabilistic features. Behavioral equivalence is formalized as a contextual, or testing, equivalence, and two main lines of research are pursued in the thesis.

The first part of the thesis focuses on contextual equivalence as a way of investigating the expressiveness of different languages. The discriminating powers offered by higher-order concurrent languages (Higher-Order π-calculi) are compared with those offered by higher-order sequential languages (à la λ-calculus) and by first-order concurrent languages (à la CCS). The comparison is carried out by examining the contextual equivalences induced by the languages on two classes of first-order processes, namely nondeterministic and probabilistic processes. As a result, the spectrum of the discriminating powers of several varieties of higher-order and first-order languages is obtained, both in a nondeterministic and in a probabilistic setting.

The second part of the thesis is devoted to proof techniques for contextual equivalence in probabilistic λ-calculi. Bisimulation-based proof techniques are studied, with particular focus on deriving bisimulations that are fully abstract for contextual equivalence (i.e., coincide with it). As a first result, full abstraction of applicative bisimilarity and similarity are proved for a call-by-value probabilistic λ-calculus with a parallel disjunction operator. Applicative bisimulations are however known not to scale to richer languages. Hence, more robust notions of bisimulations for probabilistic calculi are considered, in the form of environmental bisimulations. Environmental bisimulations are defined for pure call-by-name and call-by-value probabilistic λ-calculi, and for a (call-by-value) probabilistic λ-calculus extended with references (i.e., a store). In each case, full abstraction results are derived.

Acknowledgments

I am deeply grateful to my advisor, Davide Sangiorgi, for constantly supporting me and for giving me great opportunities to learn and to grow as a researcher. He has always been available to patiently answer my questions, to suggest enlightening solutions, and to share and discuss new ideas. I owe him a lot for all the time he devoted to me.

I want to express my gratitude to Christel Baier and Eijiro Sumii, for accepting to review this thesis and for providing insightful comments and suggestions.

In the past three years, I had the pleasure to work with many inspiring people, whom I thank. In Bologna, I had enriching discussions and collaborations with Ugo Dal Lago, Marco Bernardo, and Raphaëlle Crubillé. In Paris, I had the chance to work with Catuscia Palamidessi and Kostas Chatzikokolakis, and to learn about exciting new topics.

The work in this thesis is also the result of the opportunities I had to discuss it in different venues. In particular, I am grateful to Lars Birkedal, Luke Ong, and Damien Pous for inviting me to present my work.

I greatly profited from discussions with people in the Focus team, and with past and present colleagues. Thank you all. I especially want to thank Francesco Gavazzo, Jean-Marie Madiot, and Saverio Giallorenzo, who have been helpful friends and PhD fellows.

Contents

List of Figures

Chapter 1

Introduction

Program equivalence is a delicate notion. Nevertheless, there is a unifying and general way of defining what it means for two systems to be equivalent with respect to their behavior. This is given by the so-called contextual, or testing, equivalences: contexts of some language play the role of tests, and two programs or systems are contextually equivalent if the execution of the same test returns the same observation. More formally, given two systems S_1 and S_2 (the tested systems), a language $\mathcal{L}$ that we can use to interact with the systems (the testing language), and an observation Obs (the result of the tests), we say that S_1 and S_2 are contextually equivalent in $\mathcal{L}$ if whenever we put them in the same context C of $\mathcal{L}$ we have $Obs(C[S_1]) = Obs(C[S_2])$. This definition of contextual equivalence thereby formalizes the idea of behavioral equivalence as interchangeability, or indistiguishability, in a black-box testing scenario.

The study of contextually-defined equivalences has been pursued along different paths:

- analyzing the discriminating power of a language, as compared to other languages. In this case, we are studying the *expressiveness* of a language by looking at the kind of tests a context of the language can perform;

- studying methods allowing us to prove that two programs in a language are contextually equivalent (with respect to the same language). This approach is thereby devoted to finding *proof techniques* for contextual equivalence.

The first perspective has been adopted in particular in concurrency theory, in which several varieties of testing scenarios have been proposed. Given a class $\mathcal{C}$ of tested systems, we look at how one or more languages interact with systems in $\mathcal{C}$. Then language $\mathcal{L}_1$ is strictly more discriminating than language $\mathcal{L}_2$ if whenever S_1 and S_2 are equivalent with respect to tests (or contexts) in $\mathcal{L}_1$ then they are also equivalent with respect to tests in $\mathcal{L}_2$, and there are systems that can be discriminated by $\mathcal{L}_1$ but are equal in $\mathcal{L}_2$.

In the second line of research, we typically consider contextual equivalences where both the tested programs and the contexts are from the same language $\mathcal{L}$. Contextual equivalence defines what it means for two programs in the language to be equivalent, and we look for efficient methods to prove program equivalence.

In both cases, however, the characterization of contextually-defined equivalences in terms of equivalences whose definition is not directly of the form “for all contexts of the language...” or “for all tests...” plays a crucial role. When studying the expressiveness of a language, we aim at characterizing the contextual equivalence it induces on some class $\mathcal{C}$

of systems as an equivalence that is directly defined on the systems and does not mention the testing language. This gives us a way of comparing the testing equivalences induced by different languages with each other. On the other side, if we are interested in proving contextual equivalence for a language, we can see that the universal quantification over all contexts of a language makes it hard to exhibit proofs of equivalence. This holds in particular for higher-order languages, whose operators are capable of manipulating and exchanging programs themselves. In this setting, bisimulation-based equivalences have been shown to provide efficient proof methods for contextual equivalences.

This thesis analyzes program equivalence for higher-order languages along these two main lines of research: expressiveness and proof techniques. In particular, we focus on how higher-order languages interact with probabilistic systems and features.

The theory of functional higher-order languages, starting from λ-calculi, has been thoroughly studied in the literature, and higher-order languages for concurrent and distributed systems have been investigated as well. The interest in probabilistic programming and computation has been growing for the last few years, motivated, for instance, by the need of modeling complex systems evolving with some degree of uncertainty, and by the need of implementing randomized algorithms for both efficiency and security reasons. Probabilistic languages, equivalences, and models have been thereby proposed to this end, and they now form an established and productive research topic.

In the presence of probabilities, the definition of program equivalence must take into account the quantitative information that emerges from the systems under consideration. In contextual equivalence, this information is embedded in the notion of observation, which measures the successfulness of a test. On deterministic systems, we can observe whether the execution of a test succeeds or not; if the system is nondeterministic, we can observe whether there exists a succeeding run, or whether all runs succeed. By contrast, on probabilistic systems we do not only observe the *possibility* of succeeding but also the *probability* of success.

The following sections are devoted to a general introduction to the main languages and notions studied in this thesis, and that we will formally introduce in the next chapters. We conclude with an outline of the thesis.

1.1 Higher-order calculi, concurrency, and probabilities

Formally, higher-order calculi are calculi with variables that can be replaced by terms of the language itself. We start from the λ-calculus, the core of functional higher-order languages. Then we move to process calculi and their extension with higher-order features or probabilistic features.

1.1.1 λ-calculi

The λ-calculus [Bar84] is the paradigmatic example of higher-order calculus, in that it is a pure calculus of higher-order functions. Every term of the language represents a function, and the only operation allowed is β-reduction. Given a function $\lambda x.f_1$ in variable x applied to an argument f_2 (a function itself), β-reduction allows us to substitute f_2 to the free variable x in f_1.

There are different reduction strategies that we can adopt when evaluating a term of the

λ-calculus. In the call-by-name reduction strategy, when a function is applied to an argument then the argument is substituted to the variable of the function as it is. On the contrary, in the call-by value reduction strategy, the argument of a function is first reduced to a value (i.e., a function that cannot be further reduced) and then substituted.

The λ-calculus is at the core of functional programming languages, and many extensions with computational effects have been considered. To take into account features of imperative programming languages, λ-calculi can be extended with references and a store, as in ML-like languages [MTHM97]. Other computational effects for λ-calculi concern nondeterminism and probabilities. One of the easiest way to obtain such extensions consists in adding to the pure λ-calculus a binary choice operator $\oplus$. In the nondeterministic case, the choice between term M and term N is the term $M \oplus N$ that nondeterministically reduces either to M or to N [Ong93; San94; dP95]. In the probabilistic case, $\oplus$ denotes a choice with uniform probability, i.e., $M \oplus N$ reduces with one half probability to term M and with one half probability to term N [DZ12]. Hence, the result of the evaluation of a term is a probability distribution on functions. An analogous solution in the probabilistic case consists in endowing the choice operator with a probability value p, where $M \oplus_p N$ denotes the program that with probability p is M and with probability $1 - p$ is N [Jon90]. Indeed, several varieties of functional higher-order languages with probabilistic operators have been introduced, from abstract ones [SD78; RP02; PPT08] to more concrete ones [Pfe01; Goo13], also considering continuous distributions [BDGS16; SYWHK16].

1.1.2 Process calculi and models

In concurrent systems, we have multiple programs running in parallel. So, we can use processes rather than functions as modeling tools, since the latter ones are more suitable for representing sequential computations.

The process calculus CCS (Calculus of Communicating Systems) was first introduced by Milner in [Mil80], and its theory was further developed in [Mil89]. It is a language with operators for parallel composition and nondeterministic choice, whose semantics is formalized by means of labeled graphs (Labeled Transition Systems). These structures are nondeterministic and have labels allowing us to represent interactions between processes: we can think of labels as communication channels, on which processes can synchronize.
Higher-order concurrency combines functional programming and concurrent programming: the ability of exchanging values, common in concurrency, is enhanced by allowing values to include terms of the language itself, the distinguishing feature of functional languages. Calculi of this kind include CHOCS [Tho93] and the Higher-Order π-calculus [San92], which is the extension of CCS with higher-order features. CCS is a *first-order* concurrent language, since communication in CCS is just synchronization on atomic (first-order) input-output channels. By contrast, communication in HOπ has a more complex structure. When a process communicates on an output channel, it sends in output a process. A process with the same input channel can then synchronize with the output channel and receive the process that was sent. This communication is higher-order, since it is a process (i.e., a term of the calculus) that is exchanged in the communication.

An important extension to concurrent higher-order languages concerns distribution.

This is usually achieved by means of constructs for expressing and operating on locations. As a consequence, the observable behavior of a system of processes depends not only on the behavior of the constituent processes, but also on the locations in which these processes are run. This can have a deep impact on the behavioral theory and algebraic laws for the language. One of the simplest constructs that show these phenomena is *passivation*. Passivation offers the capability of capturing the content of a certain location, and then restarting the execution in different contexts. The semantics of passivation has been the subject of a number of papers, usually in extensions of the Higher-Order π-calculus [LSS09a; LSS09b; LSS11; PS12a; KH13]. Passivation is also featured in the Homer calculus [GH05] and the M-calculus [SS03]; a similar construct appears in the Seal calculus [CVN05] and in Acute [Sew+07]. Passivation has been advocated to support run-time system updates, fault recovery and fault tolerance (by providing the basis for mechanisms for checkpointing computations and replicating them), and to support adaptive behaviors.

As far as probabilistic extensions of process calculi are concerned, CCS with a probabilistic binary choice operator and its semantics have been investigated, e.g., in [YL92], and with a different semantics in [DD07] and [Hen12]. Since the behavior of processes running in parallel is nondeterministic, the processes represented in probabilistic extensions of CCS have both nondeterministic and probabilistic choices.
A strict subset of this class of processes is that of reactive probabilistic processes (also known as Markov decision processes or labeled Markov chains) which have, besides probabilistic choices, only a limited form of nondeterminism, i.e., external nondeterminism. External nondeterminism is a choice between different transitions with different labels and represents choices that can be made by an external user interacting with the process. By contrast, internal nondeterminism is a choice between transitions labeled by the same action and represents choices that are internally made by the system. The classical parallel operator of CCS is not closed with respect to this class of processes, hence process algebras with a parametrized parallel operator have been proposed. See [SV04] for an overview of probabilistic process algebras and classes of probabilistic processes.
Probabilistic extensions of higher-order process calculi have not been proposed yet.

1.2 Equivalence of programs

Section 1.2.1 is devoted to bisimulations for nondeterministic and probabilistic processes. Bisimulations for higher-order languages are presented in Section 1.2.3, after discussing testing and contextual equivalences (Section 1.2.2).

1.2.1 Behavioral equivalences on processes

It is not easy to understand what it means for two processes to have the same behavior. If we are only interested in the behavior of the systems, requiring the structures of the processes to be isomorphic is too strong a condition. At the same time, many equivalence relations defined in the literature might be too under-discriminating when applied to nondeterministic processes ([Gla01] compares several varieties of equivalence relations on processes). Trace-based equivalences, for instance, identify two processes by comparing the sequences of actions they can (or cannot) perform. Hence, they are not sensitive to the branching-time of processes.

Bisimulation relations independently appeared in modal logic, in computer science and in set theory between the 1970s and the 1980s [San12b], respectively in the works by van Benthem [Ben83] on the expressiveness of propositional modal languages and classical first-order languages, in the works of Milner [Mil80; Mil89] and Park [Par81] on the semantics of interactive systems, and in the works by Aczel on non well-founded sets [Acz88]. Bisimulations induce an equivalence relation on processes, i.e., bisimilarity, which is taken to be a suitable notion of behavioral equivalence on Labeled Transition Systems. According to bisimilarity, processes P and Q are equivalent if whenever P can perform an action then Q can mimic the same action and the reached states are still equivalent, and vice-versa. Furthermore, bisimilarity has a simple proof method: in order to prove that two processes are equivalent, we exhibit a relation containing the pair of processes and we verify that the relation is a bisimulation. This holds because bisimilarity is a coinductive relation, whose definition rests on the dual of the induction principle and allows for a form of circularity. See [San12a] for a fixed-point approach to coinduction and [JR12] for a (co)algebraic approach.

Probabilistic bisimulation was first proposed in [LS91], for reactive probabilistic systems. This bisimulation takes into account the quantitative information that is now available in the underlying structures it is applied to, by considering not only the possibility but also the probability of performing a state-transition. The definition was extended to processes with both probabilities and nondeterminism in [SL95; Seg95]. In recent years, several varieties of definitions and characterizations of probabilistic bisimulation and coarser probabilistic equivalences have been studied [DD11; Hen12; BDL14a; Den14].

1.2.2 Contextual and testing equivalence

Contextual equivalence was first defined by Morris in [Mor68] for the pure λ-calculus. Terms M and N are contextually equivalent if for any context C (i.e., for any term of the language with a hole), term $C[M]$ (denoting the substitution of M to the hole of C) converges (i.e., reduces to a value) if and only if $C[N]$ does.
For the nondeterministic λ-calculus, we observe the existence of a reduction sequence that converges, or, in other words, the possibility of convergence; for the probabilistic λ-calculus [DLSA14; CD14] the observability predicate is the probability of convergence of a term.

On first-order process algebras, different formulations of contextual equivalence have been examined. *May testing equivalence* [DH84; BDP99] is a contextual equivalence on process algebras where the observability predicate holds if there exists an internal computation that reaches a successful state, i.e., a state that can perform an action denoting success (corresponding to convergence in pure λ-calculi). Analogously, the observability predicate of *must testing equivalence* holds if all the internal computations succeed. On CCS-like languages, however, testing equivalences correspond to trace-based equivalences [DH84; Phi87]. In order to have a contextual equivalence for CCS that coincides with bisimilarity, we have to consider *barbed congruence* [MS92], that is, a bisimulation-based contextual equivalence where the observability predicate is the set of actions allowed from a state (its *barbs*) and the bisimulation game is only played on internal reductions. Barbed congruence for HOπ is studied in [San92].

In the probabilistic case, may and must testing preorders for process algebras have been studied in [DGHMZ07a; DGHM09] for the process algebra pCSP, and have been proved to coincide with the probabilistic simulation preorder and the probabilistic failure simulation preorder, respectively. In [DD07] and [Hen12], probabilistic barbed congruence is defined and it is shown that, analogously to the nondeterministic case, it coincides with probabilistic bisimilarity on nondeterministic and probabilisitic processes.

Testing equivalences are defined in a general form by Abramsky in [Abr87]. A testing equivalence is determined by a set of tested systems, a set of tests, a mechanism assigning an output to the application of a test, and an observability predicate on the class of outputs. The same paper focuses on defining a language of tests (that resemble logical formulas, since they have explicit conjunction, disjunction and quantifiers) that allows us to recover bisimilarity on nondeterministic processes as a testing equivalence. On reactive probabilistic processes, characterizations of bisimulation as a testing equivalence via "logical" tests have been proposed in [LS91] and [BMOW05], showing how a smaller class of tests is sufficient in order to recover probabilistic bisimilarity in this case.

1.2.3 Bisimulations for higher-order languages

Due to the universal quantification on the contexts of the language, it is generally hard to prove that two terms are contextually equivalent. Contextual equivalence proofs are particularly hard to carry out if the language under consideration has higher-order features. Bisimulations offer an efficient, operational proof method; it is therefore desirable to find bisimulation relations which are sound with respect to contextual equivalence, i.e., bisimulations inducing an equivalence relation - bisimilarity - that implies contextual equivalence. Ideally, bisimilarity should be fully abstract with respect to contextual equivalence, i.e., coincide with it.

Applicative bisimilarity [Abr90] is such an equivalence relation, reflecting the standard definition of extensional equivalence for functions. Two λ-terms M and N are applicative bisimilar if whenever M reduces to function $\lambda x.M'$, N reduces to a function $\lambda x.N'$ such that for any term P given as input to the functions we still have equivalent terms $M\{P/x\}$ and $N\{P/x\}$. Applicative bisimilarity coincides with contextual equivalence both in the call-by-name and in the call-by-value λ-calculus, while it is only sound with respect to (and does not coincide with) contextual equivalence in the call-by-name and the call-by-value nondeterministic λ-calculi [Ong93; Las98; Pit12]. The same result holds for probabilistic applicative bisimilarity in the call-by-name probabilistic λ-calculus [DLSA14], while in the call-by-value case completeness is recovered, and thus probabilistic applicative bisimilarity is fully abstract [CD14].

Applicative bisimilarity has a simple definition, but it also has two main drawbacks. First, the proof of congruence (the property that in turn allows us to prove soundness) is carried out by exploiting a sophisticated and hard to scale technique called "Howe's method" [How89; How96; Pit12]. Then, as argued in [KLS11], in calculi with features such as local store, exceptions, generative names, or existential types, and more generally in calculi with forms of information hiding, applicative bisimulation is not sound and we need to resort to bisimulations equipped with a notion of environment. *Environmental bisimulations* [SKS11], refining earlier proposals in [BS98; AG98; JR99; SP07a; KW06b], address the two problems illustrated above. Intuitively, the environments collect the observer's knowledge about values computed during the bisimulation game. The elements of the environment can then be used to construct terms to be supplied as inputs during the bisimulation game. The notion has been applied to a variety of languages, including pure λ-calculi [SP07b; SKS11], extensions of λ-calculi [SP07a; KW06b; KW06a; BL13; ABLP16], and languages for concurrency or distribution [SS09; PS11; PS12a].

1.3 Outline of the thesis

This thesis is divided into two parts, reflecting the two lines of research for contextual equivalences discussed in the introduction.

- Part I, "Discriminating power via testing equivalences", compares the expressiveness of different calculi and models, from higher-order ones to first-order ones, by considering them as testing languages that are applied to discriminating both nondeterministic systems and probabilistic systems.
- Part II, "Full abstraction for probabilistic λ-calculi", studies coinductive proof techniques for λ-calculi with a probabilistic choice operator. In particular, the problem of defining relations that are fully abstract with respect to contextual equivalences or preorders in extended lambda calculi is addressed.

Part I is based on works published in [BSV14a] and [BSV14b]. Both works are co-authored with Marco Bernardo and Davide Sangiorgi. The material in Part II has been published in [CDLSV15], co-authored with Raphaëlle Crubillé, Ugo Dal Lago, and Davide Sangiorgi, and [SV16], co-authored with Davide Sangiorgi. These papers are briefly summarized in Sections 1.3.1 and 1.3.2.

Each of the two parts of the thesis is composed as follows. First, we review the relevant background. Then we present our contributions (each chapter corresponds to a revised and extended version of the published works). Finally, we conclude and discuss additional related work and future work.

1.3.1 Discriminating power via testing equivalences

In [BSV14a], the discriminating powers of a number of higher-order languages are analyzed and compared. Both higher-order sequential languages (i.e., λ-calculi) and higher-order concurrent languages (i.e., Higher-Order π-calculi) are considered, and they are compared to first-order process calculi (CCS-like) as well. The comparison is carried out by using the languages to execute tests, formalized as contexts of the language, on first-order processes. The tests are first applied to nondeterministic processes and then to reactive probabilistic processes. The purpose of the paper is twofold:

- to compare the discriminating powers of the languages with respect to the same class of processes (the class of nondeterministic processes first, then the class of reactive probabilistic processes), and to characterize the contextual equivalences induced by the languages as known behavioral equivalences;
- to compare the discriminating power of a language on nondeterministic processes to that of the same language on probabilistic processes, highlighting some cases in which the interplay between higher-order or concurrent features and probabilities increases the discriminating power of a language.

In [BSV14b], testing equivalences on reactive probabilistic processes are analyzed, by considering three classes of first-order tests: nondeterministic processes, reactive probabilistic processes and processes featuring both (full) nondeterminism and probabilistic choices. These classes of tests are proved to have different discriminating powers, and their position in the spectrum of equivalences for reactive probabilistic processes is studied.

1.3.2 Full abstraction for probabilistic λ-calculi

In [CDLSV15], a call-by-value probabilistic λ-calculus endowed with Plotkin's disjunction operator (or "parallel or") is considered. The paper proves that not only applicative bisimilarity is fully abstract with respect to contextual equivalence (i.e., it coincides with it, being both sound and complete), but also the applicative simulation preorder is fully abstract with respect to the contextual preorder in this calculus. The latter result was known not to hold without the disjunction operator [CD14].

In [SV16], environmental bisimulations for probabilistic λ-calculi are defined, so as to have a proof technique applicable to probabilistic calculi with effects such as a local store. In order to achieve full abstraction of environmental bisimilarity, some non-trivial modifications to the definition of environmental bisimulations for non-probabilistic calculi are required:

- in probabilistic calculi a term might evaluate (even with probability one) in a non-finite number of steps. Thus, the bisimulation game is played with big-step, infinitary reductions;

- in order to have full abstraction, we are forced to define the bisimulation game directly on probability distributions on values;

- we must distinguish between different forms of environment, depending on the language we are considering.

The paper shows that bisimulations built by taking into account these three new features are fully abstract for contextual equivalence for call-by-name, call-by-value, and imperative (with a higher-order, local store) probabilistic λ-calculi.

Part I

Discriminating power via testing equivalences

Chapter 2

Background

We introduce three models for first-order processes: nondeterministic processes, formalized as LTSs, probabilistic and nondeterministic processes (NPLTSs), and reactive probabilistic processes (RPLTSs). We recall a number of behavioral equivalences for these first-order processes, the relations between them, and some important alternative characterizations of the equivalences. We conclude by recalling the language and semantics of pure λ-calculi.

2.1 Nondeterministic and probabilistic models

The behavior of a (fully) nondeterministic process can be represented through a labeled transition system.

Definition 2.1. A *labeled transition system* (LTS) is a triple $(S, \mathcal{A}, \longrightarrow)$ where S is a countable set of states (usually called processes), $\mathcal{A}$ is a countable set of transition-labeling actions, and $\longrightarrow \subseteq S \times \mathcal{A} \times S$ is a transition relation. The LTS is *image-finite* if $\{s' \in S \mid s \xrightarrow{a} s'\}$ is finite for all $s \in S$ and $a \in \mathcal{A}$.

We can generalize LTSs to more expressive models, that admit both nondeterministic and probabilistic choices.

Definition 2.2. A *nondeterministic and probabilistic labeled transition system*, NPLTS for short, is a triple $(S, \mathcal{A}, \longrightarrow)$ where S is a countable set of states, $\mathcal{A}$ is a countable set of transition-labeling actions, and $\longrightarrow \subseteq S \times \mathcal{A} \times \mathcal{D}(S)$ is a transition relation, with $\mathcal{D}(S)$ being the set of discrete probability distributions over S.

We denote probability distributions by Δ, Θ.... We can represent an LTS as an NPLTS where all distributions are Dirac distributions $\texttt{dirac}(s)$, i.e., distributions assigning probability one to a single state. Formally: $\texttt{dirac}(s)(s) = 1$ and $\texttt{dirac}(s)(s') = 0$ for all $s' \in S \setminus \{s\}$.

In any state of an NPLTS, like in LTSs, nondeterministic choices can be both internal (i.e., multiple transitions each with the same label) and external (i.e., multiple transitions each with different labels). A reactive probabilistic process features external nondeterministic choices, probabilistic choices, but no internal nondeterminism. In other words, we can see a RPLTS as a model where the choice of the action to be performed is made by the external environment, and then the target state is selected internally but purely probabilistically. Its behavior can be described as a variant of an NPLTS.

Definition 2.3. A *reactive probabilistic labeled transition system* (RPLTS) is an NPLTS $(S, \mathcal{A}, \longrightarrow)$ such that $s \xrightarrow{a} \Delta_1$ and $s \xrightarrow{a} \Delta_2$ imply $\Delta_1 = \Delta_2$ for all $s \in S$ and $a \in \mathcal{A}$.

2.2 Behavioral equivalences for nondeterministic processes

We introduce several varieties of behavioral equivalences for nondeterministic processes, from the coarser ones (trace-based equivalences) to the finer ones (simulation-based and bisimulation equivalences).

In Chapter 3, we will characterize the contextual equivalences induced on LTSs in terms of simulation equivalence [Mil89], ready simulation equivalence [BIM95; LS91], trace equivalence [BHR84], failure equivalence [BHR84], and failure-trace equivalence [Phi87; Gla01].

2.2.1 Decorated traces and sets

Definition 2.4. Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an LTS and $s, s' \in S$. The sequence $c \overset{\texttt{def}}{=} s_0, s_1 \dots s_{n-1}, s_n$ is a *computation* of $\mathcal{L}$ of length n from $s = s_0$ to $s' = s_n$ labeled by $\sigma = a_1, a_2 ...., a_n$ if for all $i = 1, \dots, n$ there exists a transition $s_{i-1} \xrightarrow{a_i} s_i$. We denote by $\mathcal{C}_{\text{fin}}(s)$ the set of finite-length computations from s.

Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an LTS and $s, s_1, s_2 \in S$. We define the following sets of computations:

- $\mathcal{C}(s, \sigma)$ is the set of computations from s labeled with trace $\sigma \in \mathcal{A}^*$ (the finite sequences of actions in $\mathcal{A}$).

- $\mathcal{CC}(s, \sigma)$ is the set of completed computations from s labeled with $\sigma \in \mathcal{A}^*$, i.e., the computations from s labeled with σ and such that the last state of the computation cannot perform any other transition.

- $\mathcal{FC}(s, \varphi)$, where $\varphi = (\sigma, F) \in \mathcal{A}^* \times 2^{\mathcal{A}}$ is a failure pair, is the set of computations from s labeled with σ such that the last state of each computation cannot perform any action in F.

- $\mathcal{RC}(s, \varrho)$, where $\varrho = (\sigma, R) \in \mathcal{A}^* \times 2^{\mathcal{A}}$ is a ready pair, is the set of computations from s labeled with σ such that the set of actions that can be performed by the last state of each computation is precisely R.

- $\mathcal{FTC}(s, \phi)$, where $\phi = (a_1, F_1) \dots (a_n, F_n) \in (\mathcal{A} \times 2^{\mathcal{A}})^*$ is a failure trace, is the set of computations from s labeled with $a_1 \dots a_n$ such that the state reached by each computation after the i-th step, for $1 \leq i \leq n$, cannot perform any action in F_i.

- $\mathcal{RTC}(s, \rho)$, where $\rho = (a_1, R_1) \dots (a_n, R_n) \in (\mathcal{A} \times 2^{\mathcal{A}})^*$ is a ready trace, is the set of computations from s labeled with $a_1 \dots a_n$ such that the set of actions that can be performed by the state reached by each computation after the i-th step, for $1 \leq i \leq n$, is precisely R_i.

2.2.2 Equivalences and preorders

We can now define several varieties of trace-based equivalences on LTSs, one for every different kind of decorated trace.

Definition 2.5. Let $(S, \mathcal{A}, \longrightarrow)$ be an LTS. Processes $s_1, s_2 \in S$ are:

- *trace equivalent* $(s_1 \sim_{\mathrm{Tr}} s_2)$ iff $\mathcal{C}(s_1, \sigma) \neq \emptyset \iff \mathcal{C}(s_2, \sigma) \neq \emptyset$ for all $\sigma \in \mathcal{A}^*$;
- *completed trace equivalent* $(s_1 \sim_{\mathrm{CTr}} s_2)$ iff $s_1 \sim_{\mathrm{Tr}} s_2$ and $\mathcal{CC}(s_1, \sigma) \neq \emptyset \iff \mathcal{CC}(s_2, \sigma) \neq \emptyset$ for all $\sigma \in \mathcal{A}^*$;
- *failure equivalent* $(s_1 \sim_{\mathrm{F}} s_2)$ iff $\mathcal{FC}(s_1, \varphi) \neq \emptyset \iff \mathcal{FC}(s_2, \varphi) \neq \emptyset$ for all $\varphi \in \mathcal{A}^* \times 2^{\mathcal{A}}$.
- *ready equivalent* $(s_1 \sim_{\mathrm{R}} s_2)$ iff $\mathcal{RC}(s_1, \varrho) \neq \emptyset \iff \mathcal{RC}(s_2, \varrho) \neq \emptyset$ for all $\varrho \in \mathcal{A}^* \times 2^{\mathcal{A}}$.
- *failure trace equivalent* $(s_1 \sim_{\mathrm{FTr}} s_2)$ iff $\mathcal{FTC}(s_1, \phi) \neq \emptyset \iff \mathcal{FTC}(s_2, \phi) \neq \emptyset$ for all $\phi \in (\mathcal{A} \times 2^{\mathcal{A}})^*$.
- *ready trace equivalent* $(s_1 \sim_{\mathrm{RTr}} s_2)$ iff $\mathcal{RTC}(s_1, \rho) \neq \emptyset \iff \mathcal{RTC}(s_2, \rho) \neq \emptyset$ for all $\rho \in (\mathcal{A} \times 2^{\mathcal{A}})^*$.

Traced-based equivalences are inductive equivalences. By contrast, simulation-based equivalences have coinductive definitions.

Definition 2.6. Let $(S, \mathcal{A}, \longrightarrow)$ be an LTS and $\mathcal{R}$ be a binary relation over S. Relation $\mathcal{R}$ is a *simulation* if, whenever $(s_1, s_2) \in \mathcal{R}$, then for all $a \in \mathcal{A}$ it holds that for each $s_1 \xrightarrow{a} s_1'$ there exists $s_2 \xrightarrow{a} s_2'$ such that $(s_1', s_2') \in \mathcal{R}$. Relation $\mathcal{R}$ is a *ready simulation* if, additionally, $s_1 \stackrel{a}{\not\longrightarrow}$ implies $s_2 \stackrel{a}{\not\longrightarrow}$. Relation $\mathcal{R}$ is a *bisimulation* if both $\mathcal{R}$ and its inverse are simulations, i.e., whenever $(s_1, s_2) \in \mathcal{R}$, then for all $a \in \mathcal{A}$ it holds that:

- for each $s_1 \xrightarrow{a} s_1'$ there exists $s_2 \xrightarrow{a} s_2'$ such that $(s_1', s_2') \in \mathcal{R}$
- for each $s_2 \xrightarrow{a} s_2'$ there exists $s_1 \xrightarrow{a} s_1'$ such that $(s_1', s_2') \in \mathcal{R}$

Processes $s_1, s_2 \in S$ are *simulation equivalent* $(s_1 \sim_{\mathrm{S}} s_2)$ – resp., *ready simulation equivalent* $(s_1 \sim_{\mathrm{RS}} s_2)$ – if there exist two simulations – resp., ready simulations – $\mathcal{R}$ and $\mathcal{R}'$ such that $(s_1, s_2) \in \mathcal{R}$ and $(s_2, s_1) \in \mathcal{R}'$. Processes $s_1, s_2 \in S$ are *bisimilar* $(s_1 \sim_{\mathrm{B}} s_2)$, or bisimulation equivalent, if there exists a bisimulation $\mathcal{R}$ such that $(s_1, s_2) \in \mathcal{R}$.

Except for bisimilarity, it is possible to define any of the equivalences $\sim$ considered above by first taking the corresponding preorder $\lesssim$, and then defining the equivalence as the intersection of the preorder and its inverse, i.e., $\sim = \lesssim \cap (\lesssim)^{-1}$.
For trace-based equivalences, the preorders are obtained by using $\Longrightarrow$ instead of $\iff$ in the definition. For instance, the trace preorder is defined as: $s_1 \lesssim_{\mathrm{Tr}} s_2$ iff $\mathcal{CC}(s_1, \sigma) \neq \emptyset \Longrightarrow \mathcal{CC}(s_2, \sigma) \neq \emptyset$ for all $\sigma \in \mathcal{A}^*$. For the failure trace preorder and the ready trace preorder, moreover, we have to require as well that the initial states have the same ready set. This is implicit when considering equivalences, since trace equivalent states s, s' have the same ready set, but it has to be made explicit for the failure trace preorder and the ready trace preorder, since in the definition of failure trace or ready trace we do not allow

a failure set or ready set at the beginning of the trace.[1]
For simulation-based preorders, we simply require the existence of a simulation, or a ready simulation. Hence, processes $s_1, s_2 \in S$ are in the simulation preorder ($s_1 \lesssim_{\mathrm{S}} s_2$) – resp., in the ready simulation preorder ($s_1 \lesssim_{\mathrm{RS}} s_2$) – if there exists a simulation – resp., a ready simulation – $\mathcal{R}$ such that $(s_1, s_2) \in \mathcal{R}$.

2.2.3 Spectrum for LTSs

The relations between all the equivalences defined in the previous section are summarized in Figure 2.1, where arrows represent strict inclusions [Gla01].[2]

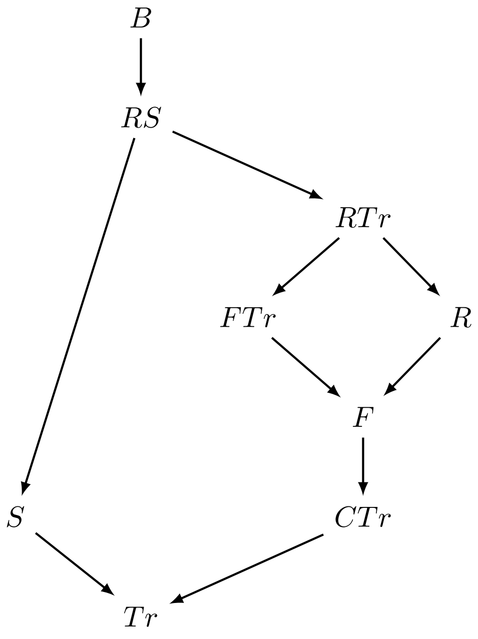

Figure 2.1: Spectrum for LTSs

2.2.4 Logical characterizations

In the following chapters, we are going to exploit the logical characterizations of the coinductive equivalences that we have defined, in particular for ready simulation equivalence and simulation equivalence. The logical characterization of bisimilarity is given by the Hennessy Milner Logic (HML). The formulas of HML are defined by the following grammar:

$$F ::= \top \;\Big|\; \neg F \;\Big|\; F_1 \wedge F_2 \;\Big|\; \langle\, a \,\rangle F$$

for a ranging over the labels in a given set $\mathcal{A}$. Given an LTS $(S, \mathcal{A}, \longrightarrow)$, we define the satisfiability of formula F in state s (notation: $s \models F$) as follows:

[1] We could have equivalently defined failure trace equivalence using the definition of failure trace that allows a failure set at the beginning of the trace as well, and analogously for ready trace equivalence.

[2] Standard counterexamples for the strictness of these inclusions will be presented in examples in Chapter 3.

$$\begin{array}{lll}
s \models \top & \text{always} & \\
s \models \neg F & \text{iff} & s \not\models F \\
s \models F_1 \wedge F_2 & \text{iff} & s \models F_1 \text{ and } s \models F_2 \\
s \models \langle a \rangle F & \text{iff} & \text{there is a } s' \text{ such that } s \xrightarrow{a} s' \text{ and } s' \models F.
\end{array}$$

Proposition 2.7 ([HM85]). *Let $(S, \mathcal{A}, \longrightarrow)$ be an image finite LTS. For every $s_1, s_2 \in S$, $s_1 \sim_{\mathrm{B}} s_2$ if and only if for every formula F of HML, $s_1 \models F$ if and only if $s_2 \models F$.*

Analogous characterization results, using weaker modal logics, hold for ready simulation equivalence and simulation equivalence. The formulas of *Ready Simulation Logic (RSL)* on $\mathcal{A}$ are defined as follows:

$$F ::= \top \;\Big|\; \neg a \;\Big|\; F_1 \wedge F_2 \;\Big|\; \langle a \rangle F$$

In contrast with Hennessy Milner Logic, this logic does not have a full negation operation $\neg F$, since negation is limited to single (terminal) actions, via the operator $\neg a$. The satisfiability of $\neg a$ is defined as:

$$s \models \neg a \qquad \text{iff} \qquad s \overset{a}{\not\longrightarrow}$$

The formulas of *Simulation Logic (SL)* are obtained by removing action negation $\neg a$ from the definition of RSL.

Proposition 2.8 ([Gla01]). *Let $(S, \mathcal{A}, \longrightarrow)$ be an image-finite LTS. For every $s_1, s_2 \in S$,*

1. *$s_1 \sim_{\mathrm{RS}} s_2$ if and only if for every formula F of RSL, $s_1 \models F$ if and only if $s_2 \models F$*
2. *$s_1 \sim_{\mathrm{S}} s_2$ if and only if for every formula F of SL, $s_1 \models F$ if and only if $s_2 \models F$*

2.3 Behavioral equivalences for probabilistic processes

2.3.1 RPLTS

We start by defining equivalences and preorders on RPLTSs.

Given a transition $s \xrightarrow{a} \Delta$, a process $s' \in S$ is not reachable from s via that a-transition if $\Delta(s') = 0$, otherwise it is reachable with probability $p = \Delta(s')$. The reachable states form the support of Δ, i.e., $\mathit{supp}(\Delta) = \{s' \in S \mid \Delta(s') > 0\}$.

In the RPLTS setting, each state-to-state step of a computation is derived from a state-to-distribution transition $s \xrightarrow{a} \Delta$.

Definition 2.9. Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an RPLTS and $s, s' \in S$. The sequence $c \overset{\mathtt{def}}{=} s_0, s_1 \ldots s_{n-1}, s_n$ is a *computation* of $\mathcal{L}$ of length n from $s = s_0$ to $s' = s_n$ labeled by $\sigma = a_1, a_2 ...., a_n$ if for all $i = 1, \ldots, n$ there exists a transition $s_{i-1} \xrightarrow{a_i} \Delta_i$ such that $s_i \in \mathit{supp}(\Delta_i)$, with $\Delta_i(s_i)$ being the execution probability of the step from s_{i-1} to s_i via action a_i conditioned on the selection of transition $s_{i-1} \xrightarrow{a_i} \Delta_i$ of $\mathcal{L}$ at state s_{i-1}. We denote by $\mathcal{C}_{\mathrm{fin}}(s)$ the set of finite-length computations from s.

Given a computation $c \in \mathcal{C}_{\mathrm{fin}}(s)$, its conditional execution probability $\mathit{prob}(c)$ can be defined as the product of the conditional execution probabilities of the individual steps of c. This notion is lifted to a set $\mathcal{C} \subseteq \mathcal{C}_{\mathrm{fin}}(s)$ of identically labeled computations by letting $\mathit{prob}(\mathcal{C}) = \sum_{c \in \mathcal{C}} \mathit{prob}(c)$.

Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an RPLTS and $s, s_1, s_2 \in S$. The definition of the sets of decorated traces from s is defined as for LTS, but using the definition of computation given above for RPLTSs. We then introduce probabilistic trace-based equivalences on $\mathcal{L}$ as follows by analogy with [JS90; HT92]. Instead of only observing the possibility of performing a (decorated) trace, we observe the probability of performing the trace.

- $s_1 \sim_{\mathrm{PTr}} s_2$ iff $\mathit{prob}(\mathcal{C}(s_1, \sigma)) = \mathit{prob}(\mathcal{C}(s_2, \sigma))$ for all $\sigma \in \mathcal{A}^*$.
- $s_1 \sim_{\mathrm{PCTr}} s_2$ iff $s_1 \sim_{\mathrm{PTr}} s_2$ and $\mathit{prob}(\mathcal{CC}(s_1, \sigma)) = \mathit{prob}(\mathcal{CC}(s_2, \sigma))$ for all $\sigma \in \mathcal{A}^*$.
- $s_1 \sim_{\mathrm{PF}} s_2$ iff $\mathit{prob}(\mathcal{FC}(s_1, \varphi)) = \mathit{prob}(\mathcal{FC}(s_2, \varphi))$ for all $\varphi \in \mathcal{A}^* \times 2^{\mathcal{A}}$.
- $s_1 \sim_{\mathrm{PR}} s_2$ iff $\mathit{prob}(\mathcal{RC}(s_1, \varrho)) = \mathit{prob}(\mathcal{RC}(s_2, \varrho))$ for all $\varrho \in \mathcal{A}^* \times 2^{\mathcal{A}}$.
- $s_1 \sim_{\mathrm{PFTr}} s_2$ iff $\mathit{prob}(\mathcal{FTC}(s_1, \phi)) = \mathit{prob}(\mathcal{FTC}(s_2, \phi))$ for all $\phi \in (\mathcal{A} \times 2^{\mathcal{A}})^*$.
- $s_1 \sim_{\mathrm{PRTr}} s_2$ iff $\mathit{prob}(\mathcal{RTC}(s_1, \rho)) = \mathit{prob}(\mathcal{RTC}(s_2, \rho))$ for all $\rho \in (\mathcal{A} \times 2^{\mathcal{A}})^*$.

The corresponding preorders can be defined by using $\leq$ instead of $=$ in the definitions above, and, for failure trace and ready trace equivalence, by requiring that the initial states have the same ready set.

To define probabilistic bisimilarity and similarity, we first define the lifting function $\mathtt{lift}() : S \times S \to \mathcal{D}(S) \times \mathcal{D}(S)$, which lifts a relation on S to a relation on distributions over S.

Definition 2.10. Given a relation $\mathcal{R}$ over a set S and $\Delta, \Theta \in \mathcal{D}(S)$, we say that $\Delta \, \mathtt{lift}(\mathcal{R}) \, \Theta$ if there is a countable index set I and probability values $\{p_i\}_{i \in I}$ such that the following holds:

- $\sum_{i \in I} p_i = 1$
- $\Delta = \sum_i p_i \cdot \mathtt{dirac}(s_i)$
- $\Theta = \sum_i p_i \cdot \mathtt{dirac}(t_i)$
- for every $i \in I$, $s_i \mathrel{\mathcal{R}} t_i$

Definition 2.11. Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an RPLTS. A binary relation $\mathcal{R}$ on S is a *probabilistic simulation* iff, whenever $(s_1, s_2) \in \mathcal{R}$, then for all $a \in \mathcal{A}$ it holds that $s_1 \xrightarrow{a} \Delta_1$ implies $s_2 \xrightarrow{a} \Delta_2$ with $(\Delta_1, \Delta_2) \in \mathtt{lift}(\mathcal{R})$. Relation $\mathcal{R}$ is a *probabilistic bisimulation* if both $\mathcal{R}$ and its inverse are probabilistic simulations. Processes $s_1, s_2 \in S$ are *probabilistic simulation equivalent* ($s_1 \sim_{\mathrm{PS}} s_2$) if there exist two simulations $\mathcal{R}$ and $\mathcal{R}'$ such that $(s_1, s_2) \in \mathcal{R}$ and $(s_2, s_1) \in \mathcal{R}'$. Processes $s_1, s_2 \in S$ are *bisimilar* ($s_1 \sim_{\mathrm{PB}} s_2$), or bisimulation equivalent, if there exists a bisimulation $\mathcal{R}$ such that $(s_1, s_2) \in \mathcal{R}$.

Many equivalent definitions of probabilistic bisimilarity have appeared in the literature. We have introduced probabilistic similarity $\sim_{\mathrm{PS}}$ and probabilistic bisimilarity $\sim_{\mathrm{PB}}$ using the notion of probabilistic lifting of a relation, as in [Den14]. This is analogous to defining simulations and bisimulations using a weight function [JL91; Seg95; Bai98]. As shown in [Seg95], the resulting bisimilarity is equivalent to the one given by the definition by Larsen and Skou in [LS91], which requires a probabilistic bisimulation to be an equivalence relation and uses equivalence classes to compare the reached probability distributions:

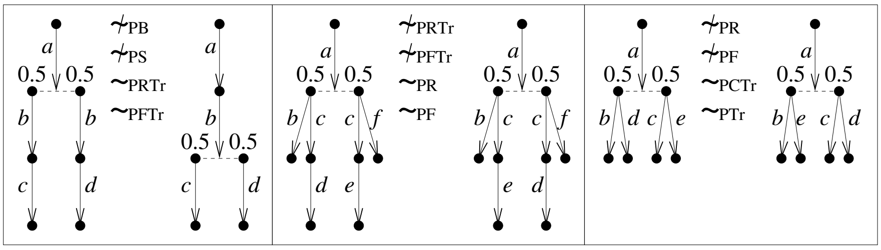

Figure 2.2: Strictness of inclusions in the spectrum for RPLTSs

An equivalence relation $\mathcal{R}$ is a probabilistic bisimulation iff $(s_1, s_2) \in \mathcal{R}$ implies that for all $a \in \mathcal{A}$ it holds that $s_1 \xrightarrow{a} \Delta_1$ implies $s_2 \xrightarrow{a} \Delta_2$ with $\Delta_1(S') = \Delta_2(S')$ for all $S' \in S/\mathcal{R}$. (PB1)

Since the relation is an equivalence, and thereby symmetric, it is not necessary to include the clause from s_2 (i.e., $s_2 \xrightarrow{a} \Delta_2$ implies $s_1 \xrightarrow{a} \Delta_1$ with $\Delta_1(S') = \Delta_2(S')$ for all $S' \in S/\mathcal{R}$). Requiring a bisimulation to be an equivalence relation is however not convenient when we want to prove that two states are bisimilar, since it requires to build a reflexive, symmetric and transitive relation.

The definitions of probabilistic bisimulation presented so far lead to the same notion of probabilistic bisimilarity. In other words, although the definitions of bisimulation do not coincide (i.e., a relation might be a bisimulation according to one definition, but not according to another one), their unions (i.e., the bisimilarities given by the different notions of bisimulation) all capture the same equivalence relation $\sim_{\text{PB}}$.

When referring to probabilistic systems, we sometimes write bisimulation instead of probabilistic bisimulation, and analogously for the other probabilistic equivalences and preorders. For decorated traces, we also sometimes omit the notation for the specific set of computations when it is clear from the context (e.g., if we are explicitly considering failure traces ϕ we write $\mathit{prob}(s, \phi)$ instead of $\mathit{prob}(\mathcal{FTC}(s, \phi))$).

2.3.2 Spectrum for RPLTSs

It was shown in [Bai98; BK00] that $\sim_{\text{PB}}$ and $\sim_{\text{PS}}$ coincide, hence the variants in between (ready similarity, failure similarity, completed similarity) collapse too. Moreover, the proofs of the results in [JS90; HT92] for fully probabilistic processes can be smoothly adapted to the RPLTS case, and also extended to deal with $\sim_{\text{PRTr}}$ and $\sim_{\text{PFTr}}$. As a consequence, we have the following spectrum under the assumption that every state has finitely many outgoing transitions, i.e., it is finitely-branching.

Proposition 2.12. *On finitely-branching RPLTS processes, it holds that:*

$$\sim_{\text{PB}} = \sim_{\text{PS}} \subsetneq \sim_{\text{PRTr}} = \sim_{\text{PFTr}} \subsetneq \sim_{\text{PR}} = \sim_{\text{PF}} \subsetneq \sim_{\text{PCTr}} = \sim_{\text{PTr}}$$

The strictness of all the inclusions above is witnessed by the counterexamples in Fig. 2.2. The graphical conventions for process descriptions are as follows. Vertices represent states and action-labeled edges represent action-labeled transitions. Given a transition $s \xrightarrow{a} \Delta$, the corresponding a-labeled edge goes from the vertex for state s to a set of vertices linked by a dashed line, each of which represents a state $s' \in \mathit{supp}(\Delta)$ and is labeled with $\Delta(s')$. The label $\Delta(s')$ is omitted when it is equal to 1, i.e., when Δ is the Dirac distribution $\texttt{dirac}(s')$.

2.3.3 Testing characterizations

On RPLTSs, probabilistic bisimilarity can be captured by considering a simple class of tests. Let $\mathbf{T}$ be the language of tests $\mathbf{t}$ defined as follows:

$$\mathbf{t} ::= \omega \;\big|\; a.\mathbf{t} \;\big|\; (\mathbf{t}_1, \mathbf{t}_2)$$

where a ranges over the labels in the action set $\mathcal{A}$ of the RPLTS. The test ω represents success, $a.\mathbf{t}$ sequentially checks whether it's possible to do a and then proceeds with test $\mathbf{t}$, and $(\mathbf{t}_1, \mathbf{t}_2)$ is the conjunctive test. Formally, given a reactive probabilistic process s, the probability of success $\Pr(\mathbf{t}, s)$ when the test $\mathbf{t}$ is executed on s is defined by structural induction on $\mathbf{t}$:

$$\begin{aligned}
&\Pr(\omega, s) = 1\\
&\Pr(a.\mathbf{t}, s) = \begin{cases} 0 & \text{if } s \overset{a}{\not\longrightarrow} \\ \sum_{s' \in \mathrm{supp}(\Delta)} \Delta(s') \cdot \Pr(\mathbf{t}, s') & \text{if } s \xrightarrow{a} \Delta \end{cases}\\
&\Pr((\mathbf{t}_1, \mathbf{t}_2), s) = \Pr(\mathbf{t}_1, s) \cdot \Pr(\mathbf{t}_2, s).
\end{aligned}$$

It holds that two processes are bisimilar if and only if, for every test of $\mathbf{T}$, they have the same probability of passing the test.

Theorem 2.13. *([BMOW05]) On reactive probabilistic processes, $s \sim_{\mathrm{PB}} s'$ iff $Pr(\mathbf{t}, s) = Pr(\mathbf{t}, s')$ for every test $\mathbf{t}$ in $\mathbf{T}$.*

The theorem above provides a further simplification of the class of tests defined by Larsen and Skou in [LS91], and proved to characterize probabilistic bisimulation. Indeed, the tests in [LS91] also contain a probabilistic negation operator $\neg a$ restricted to actions, defined as

$$\Pr(\neg a, s) = \begin{cases} 0 & \text{if } s \xrightarrow{a} \\ 1 & \text{if } s \overset{a}{\not\longrightarrow} \end{cases}$$

2.3.4 NPLTS and resolutions

The definitions of simulation and bisimulation on NPLTSs are the same as on RPLTSs (Definition 2.11). The definition of trace-based equivalences could not be applied directly to NPLTSs, since they rely on the fact that the model has no internal nondeterminism. To extend the definition to NPLTSs, we introduce resolutions, which represent RPLTSs that can be obtained by applying a scheduler (resolving the internal nondeterminism) to the NPLTS under consideration.

Definition 2.14. Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an NPLTS and $s \in S$. An NPLTS $\mathcal{Z} = (Z, \mathcal{A}, \longrightarrow_{\mathcal{Z}})$ is a *resolution* of s if there exists a state correspondence function $corr_{\mathcal{Z}} : Z \to S$ such that $s = corr_{\mathcal{Z}}(z_s)$, for some $z_s \in Z$, and for all $z \in Z$ it holds that:

- If $z \xrightarrow{a}_{\mathcal{Z}} \Delta$, then $corr_{\mathcal{Z}}(z) \xrightarrow{a} \Delta'$ with $corr_{\mathcal{Z}}$ being injective over $supp(\Delta)$ and $\Delta(z') = \Delta'(corr_{\mathcal{Z}}(z'))$ for all $z' \in Z$.

- If $z \xrightarrow{a_1}_{\mathcal{Z}} \Delta_1$ and $z \xrightarrow{a_2}_{\mathcal{Z}} \Delta_2$, then $a_1 = a_2$ and $\Delta_1 = \Delta_2$.

We let $z_s^{\mathcal{Z}}$ denote the correspondent of s in a resolution $\mathcal{Z}$ of s (i.e., $s = \mathit{corr}_{\mathcal{Z}}(z_s^{\mathcal{Z}})$), and we sometimes simply write z_s if the resolution we are referring to is clear from the context.

$\mathcal{Z}$ is *maximal* if, for all $z \in Z$, whenever z has no outgoing transitions, then $\mathit{corr}_{\mathcal{Z}}(z)$ has no outgoing transitions either. We respectively denote by $\mathit{Res}(s)$ and $\mathit{Res}_{\max}(s)$ the sets of resolutions and maximal resolutions of s.

As $\mathcal{Z} \in \mathit{Res}(s)$ is fully probabilistic (i.e., it has no nondeterminism), the probability $\mathit{prob}(c)$ of executing $c \in \mathcal{C}_{\text{fin}}(z_s)$ is the product of the (no longer conditional) execution probabilities of the individual steps of c. This notion is lifted to $\mathcal{C} \subseteq \mathcal{C}_{\text{fin}}(z_s)$ by letting $\mathit{prob}(\mathcal{C}) = \sum_{c \in \mathcal{C}} \mathit{prob}(c)$ whenever none of the computations in $\mathcal{C}$ is a proper prefix of one of the others.

Using resolutions, we can define trace-based equivalences on NPLTSs

Definition 2.15. Let $\sim$ be any of the trace-based equivalences defined for RPLTSs. Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an NPLTS and s, s' two states of the NPLTS. Then $s \sim s'$ if

- for every resolution $\mathcal{Z}$ of s there exists a resolution $\mathcal{Z}'$ of s' such that $z_s^{\mathcal{Z}} \sim z_{s'}^{\mathcal{Z}'}$ (where $\sim$ is defined as for RPLTSs);
- for every resolution $\mathcal{Z}'$ of s' there exists a resolution $\mathcal{Z}$ of s such that $z_s^{\mathcal{Z}} \sim z_{s'}^{\mathcal{Z}'}$ (where $\sim$ is defined as for RPLTSs).

In both items above, the second occurrence of $\sim$ is defined as for RPLTS, since resolutions are indeed RPLTS. However, since also external nondeterminism is resolved by resolutions, decorated traces would not be visible. Hence, we assume that failure or ready sets are checked on the states of the original NPLTS. For instance, given a state s in a NPLTS and a resolution $\mathcal{Z}$ of s, the probability of z_s having the failure trace $(a_1 a_2 ... a_n, F)$ is the sum of the probabilities of all the computations $z_s, z_1, z_2, ..., z_n$ from z_s in the resolution such that the computation is labeled by trace $a_1 a_2 ... a_n$ and the correspondent $\mathit{corr}_{\mathcal{Z}}(z_n)$ of the last state of the computation in the original NPLTS cannot perform any of the actions in F.

The spectrum of equivalences for NPLTS and variations over the definition of resolutions (e.g., by considering probabilistic, instead of nondeterministic, schedulers) can be found in [BDL14a; BDL13]. Finally, the probabilistic equivalences considered in this thesis are *exact* probabilistic equivalences, i.e., we require the observed probabilities to be the same in the compared processes; different approaches allow the probabilities to differ up to some bound p [DLT08].

2.4 Calculi

The terms of pure λ-calculi are generated by the following grammar:

$$M, N ::= x \;\Big|\; \lambda x.M \;\Big|\; MN$$

where x is a variable from a countable set of variables, $\lambda x.M$ is an *abstraction* and term MN is the *application* of term M to N. A term M is closed if every variable x occurring in M is bound by λx. We identify α-convertible terms. We write $M\{{}^{N}\!/\!_{x}\}$ for the capture-avoiding substitution of N for x in M. The *values* are the terms of the form $\lambda x.M$. We use meta-variables $M, N \ldots$ for terms, and $V, W, \ldots$ for values. When we write terms of the λ-calculus, we use the standard notational convention for parenthesis: abstraction binds to the right and application binds to the left.

A *context* C is an expression obtained from a term by replacing some subterms with *holes* of the form $[\cdot]_i$. We write $C[M_1, \ldots, M_n]$ for the term obtained by replacing each occurrence of $[\cdot]_i$ in C with M_i. Note that a context may contain no holes, and therefore any term is a context. The context may bind variables in $M_1, \ldots, M_n$; for example, if $C = \lambda x.[\cdot]_1$ and $M = x$, then $C[M]$ is $\lambda x.x$, not $\lambda y.x$. The indexing of the holes in contexts is usually omitted.

In call-by-name, term M reduces in one step to term N if there is a derivation of $M \longrightarrow N$ using the rules Beta-CBN and EvCon in Figure 2.3, using the CBN evaluation contexts. Evaluation contexts, in contrast with standard contexts, may have only one occurrence of a single hole $[\cdot]$. In call-by-value, one-step reduction is defined analogously, but using rule Beta-CBV and rule EvCon with the CBV evaluation contexts.

The rules are defined using a single-step (or small-step) reduction relation. We write $\Longrightarrow$ for the multi-step reduction relation, defined as the reflexive and transitive closure of $\longrightarrow$.

We use a tilde to denote a tuple; for instance, $\widetilde{M}$ is a tuple of terms $M_1, ..., M_n$ for some n, and $(\widetilde{M})_i$ is its i-th element. Hence, we write $C[\widetilde{M}]$, with $\widetilde{M} = M_1, ..., M_n$, for $C[M_1, ..., M_n]$. Sometimes we write tuples as $\{M_i\}_i$ when we want to emphasize the indexing set. All notations are extended to tuples componentwise.

We use $\lambda.M$ to denote a *thunked* term, i.e., a term $\lambda x.M$ for x a variable not occurring in M.

$$\textsc{Beta-CBN}\ \frac{}{(\lambda x.M)N \longrightarrow M\{{}^{N}\!/\!_{x}\}} \qquad \textsc{Beta-CBV}\ \frac{}{(\lambda x.M)V \longrightarrow M\{{}^{V}\!/\!_{x}\}}$$

$$\textsc{EvCon}\ \frac{M \longrightarrow N \quad C \text{ is an evaluation context}}{C[M] \longrightarrow C[N]}$$

$$\begin{array}{lll} \text{CBN evaluation contexts} & C = [\cdot] & \mid\ CM \\ \text{CBV evaluation contexts} & C = [\cdot] & \mid\ CM \ \mid\ VC \end{array}$$

Figure 2.3: Operational semantics for pure λ-calculi

Chapter 3

The discriminating power of higher-order languages

In this chapter we study the discriminating power offered by *higher-order concurrent* languages, and contrast this power with those offered by *higher-order sequential* languages (which are deprived of all concurrency) and by *first-order concurrent* languages (which are deprived of all higher-order features). The comparison is carried out by considering embeddings of first-order processes into the languages, and then examining the equivalences induced by the resulting contextual equivalences on the first-order processes. In other words, the discriminating power of a language refers to the existence of appropriate contexts of the language that are capable of separating the behaviors of first-order processes.

The higher-order sequential languages are λ-calculi with a store location, akin to imperative λ-calculi. The λ-calculi offer constructs for reading the content of the location, overriding it, and for performing basic observations on the process stored in the location. The higher-order concurrent languages are HOπ, which allows higher-order communication, and HO$\pi_{\mathtt{pass}}$, an extension of HOπ with passivation (similar to the languages in [PS12a; KH13; LSS11]). Both languages also admit first-order communications, to be able to interact with the embedded first-order processes. The first-order concurrent language that we consider is CCS$^-$, a CCS-like calculus.

The λ-calculi also allow us to observe the inability for a process to perform a certain action. In concurrency, this possibility is referred to as *action refusal*. For a thorough comparison, we therefore also consider both restrictions of the λ-calculi without the refusal observation (though at the price of allowing computations that may get stuck) and extensions of HOπ, HO$\pi_{\mathtt{pass}}$, and CCS$^-$ with the refusal capability.

Concerning the tested first-order processes, embedded into the above languages, we consider both ordinary LTSs and RPLTSs. We show that, on LTSs, the difference between the discriminating power of HOπ and HO$\pi_{\mathtt{pass}}$ is captured, in the λ-calculi, through the difference between the call-by-name and call-by-value evaluation strategies, both with and without refusals. The correspondence between the HO$\pi_{\mathtt{pass}}$ calculi and the call-by-value λ-calculi appears robust, and is maintained in all scenarios examined. The same does not hold between the HOπ calculi and the call-by-name λ-calculi, whose correspondence breaks on RPLTSs. The case of RPLTSs is more involved also when we consider the first-order language CCS$^-$. For instance, the discriminating power of CCS$^-$ is strictly in between that of the call-by-name λ-calculus and HOπ. In contrast, the three languages are equally discriminating on LTSs.

We also discuss variations of the above settings. In languages with locations, communication may or may not be affected by spatial proximity. This is the difference between global vs. local communications. This difference is important for the semantics of the languages but, as we shall see, does not impinge on their discriminating power.

The contextual equivalences that we consider are may-like (a test, i.e., a context, is successful on a process if there is at least one successful computation). We also discuss the contextual preorders, and 'must' forms of success (all computations are successful). We isolate a few scenarios in which, surprisingly, the may and must forms of contextual equivalence coincide.

Section 3.2 considers the embeddings of LTSs and RPLTSs into λ-calculi. Section 3.3.1 shows the syntax and operational semantics of the concurrent languages (CCS- and HOπ-like), whose discriminating power is studied in Sections 3.4 and 3.5. Section 3.6 discusses variations of the scenarios examined, such as language extensions and must-equivalences.

Notation: In examples, we sometimes use a CCS-like notation, with prefixing and choice, to describe the processes of an LTS or RPLTS.

3.1 Contextual equivalences

Given a set of processes as states of an LTS or RPLTS $\mathcal{L}$ and an algebraic language $\texttt{AL}$ (i.e., generated by a grammar), we can embed the states in the grammar by first taking a bijection f from the set of states s, s'... to a set of constants P, P'... added to the language, and then defining the behavior of the constants as corresponding to the behavior of the states, i.e., $s \xrightarrow{a} s'$ if and only if $f(s) = P \xrightarrow{a} P' = f(s')$. Then we say that the equivalence induced by $\texttt{AL}$ equates the $\mathcal{L}$ processes s_1 and s_2 if $C[f(s_1)]$ and $C[f(s_2)]$ behave the same for all contexts C of $\texttt{AL}$.

Here 'behave the same' is formalized as in ('may') contextual equivalence: for any P_1, P_2, $C[P_1]$ is as successful as $C[P_2]$ with respect to a special success observation, indicated with ω. The context C is an $\texttt{AL}$-expression with a *single* occurrence of the hole $[\cdot]$ in it.

We use P, Q... to range over the (constants for) processes in the language, corresponding to $\mathcal{L}$ processes s, t.... For simplicity, $\mathcal{L}$ is used both to denote the LTS or RPLTS of tested first-order processes and to denote the corresponding constants embedded in the language.

Moreover, in these tested processes each transition represents a visible action, i.e., there is a corresponding coaction with which the action can synchronize and produce a reduction; the actions available for $\mathcal{L}$ do not include the success signal ω. We write $\texttt{AL}(\mathcal{L})$ for the extension of $\texttt{AL}$ with the (constants corresponding to) $\mathcal{L}$ processes

In a language $\texttt{AL}(\mathcal{L})$, reductions are represented as τ-transitions $\xrightarrow{\tau}$ (or simply $\longrightarrow$, in λ-calculi). Each language $\texttt{AL}$ used will have constructs for testing the action capabilities of $\mathcal{L}$ processes; thus, the set of action names for $\mathcal{L}$ is supposed to appear in the grammar for $\texttt{AL}$. We emphasize that probabilities may appear in the tested $\mathcal{L}$ processes, but they may *not* appear in the $\texttt{AL}$ languages that test the processes.

The operational semantics of $\texttt{AL}(\mathcal{L})$ will be based on different LTS-like models depending on the nature of $\mathcal{L}$. A finite-length computation c from a term $M \in \texttt{AL}(\mathcal{L})$ is *successful* if each step of c is labeled with τ, the last state of c can perform ω, and no preceding state of c can perform ω. We denote by $\mathcal{SC}(M)$ the set of successful computations from M. In the nondeterministic case, when $\mathcal{L}$ is an LTS, the semantic model underlying $\mathtt{AL}(\mathcal{L})$ is again an LTS.

Definition 3.1. Let $\mathcal{L}$ be an LTS, P_1 and P_2 two processes of $\mathcal{L}$, and $\mathtt{AL}$ an algebraic language. In $\mathtt{AL}(\mathcal{L})$:

- P_1 is *contextually may-less* than P_2, written $P_1 \leq^{\mathcal{L}}_{\mathtt{AL}} P_2$, if $\mathcal{SC}(C[P_1]) \neq \emptyset \implies \mathcal{SC}(C[P_2]) \neq \emptyset$ for all contexts C of $\mathtt{AL}$.
- P_1 is *contextually may-equivalent* to P_2, written $P_1 \simeq^{\mathcal{L}}_{\mathtt{AL}} P_2$, if $P_1 \leq^{\mathcal{L}}_{\mathtt{AL}} P_2$ and $P_2 \leq^{\mathcal{L}}_{\mathtt{AL}} P_1$.

In the case that $\mathcal{L}$ is an RPLTS, the definition of contextual equivalence is more involved because the semantic model underlying $\mathtt{AL}(\mathcal{L})$ is a nondeterministic and probabilistic LTS (NPLTS). Hence, we resort to resolutions (Definition 2.14).

The contextual equivalence defined below is inspired by [YL92; JY95; Seg96; DGHM08]. Intuitively, P_1 is worse than P_2 if, for all contexts C, the maximum probability of reaching success in an arbitrary maximal resolution of $C[P_1]$ is not greater than the maximum probability of reaching success in an arbitrary maximal resolution of $C[P_2]$. To correctly quantify success, we restrict ourselves to $\mathit{Res}_{\tau,\max}(C[P])$, the set of maximal resolutions obtained from $C[P]$ by forbidding the execution of actions not resulting in interactions (i.e., τ actions).

Definition 3.2. Let $\mathcal{L}$ be an RPLTS, P_1 and P_2 be two processes of $\mathcal{L}$, and $\mathtt{AL}$ be an algebraic language. We say that in $\mathtt{AL}(\mathcal{L})$:

- P_1 is *contextually may-less* than P_2, written $P_1 \leq^{\mathcal{L}}_{\mathtt{AL}} P_2$, if for all contexts C of $\mathtt{AL}$ it holds that:

$$\bigsqcup_{\mathcal{Z}_1 \in \mathit{Res}_{\tau,\max}(C[P_1])} \mathit{prob}(\mathcal{SC}(z_{C[P_1]})) \leq \bigsqcup_{\mathcal{Z}_2 \in \mathit{Res}_{\tau,\max}(C[P_2])} \mathit{prob}(\mathcal{SC}(z_{C[P_2]}))$$

- P_1 is *contextually may-equivalent* to P_2, written $P_1 \simeq^{\mathcal{L}}_{\mathtt{AL}} P_2$, if $P_1 \leq^{\mathcal{L}}_{\mathtt{AL}} P_2$ and $P_2 \leq^{\mathcal{L}}_{\mathtt{AL}} P_1$.

We sometimes abbreviate ‘contextual may equivalence’ as ‘contextual equivalence’ or even ‘may equivalence’.

3.2 λ-calculi

3.2.1 Syntax

Figure 3.1 shows the syntax of the λ-calculus with a location Λ^{loc} into which we embed the processes of a (first-order) LTS or RPLTS $\mathcal{L}$. The grammar of the language resulting from the embedding, $\Lambda^{loc}(\mathcal{L})$, has therefore the additional production

$$M := \ldots \;\big|\; P$$

$$
\begin{array}{lllll}
\text{Terms: } M ::= x & \text{(variables)} & \mid M_1 \, \underline{\texttt{seq}} \, M_2 & & \text{(sequentialization)} \\
\quad\quad\quad \mid \ \lambda x.M & \text{(functions)} & \mid \, !loc & & \text{(read)} \\
\quad\quad\quad \mid \ M_1 M_2 & \text{(applications)} & \mid loc := M & & \text{(write)} \\
\quad\quad\quad \mid \ c & \text{(constants)} & \mid \texttt{if } M_1 \texttt{ then } M_2 \texttt{ else } M_3 & & \text{(if-then-else)} \\
\quad\quad\quad \mid \ r? & \text{(action test)} & & &
\end{array}
$$

Figure 3.1: Syntax of Λ^{loc}

where P is a constant for an $\mathcal{L}$ process, as described in Section 3.1; moreover, in the action test, r is supposed to range over the actions in $\mathcal{L}$.

The evaluation of a λ-term M is defined with respect to a location containing a process. The language includes a construct $loc := M$ that evaluates M and writes the resulting value in the location, and a construct $!loc$ for reading the content of the location. The language also features sequentialization $M \, \underline{\texttt{seq}} \, N$, where M is a command (of unit type). The action-test construct $r?$ allows us to check whether the process contained in the location can perform action r.
The remaining constructs are common constructs of typed λ-calculi. We assume that the set of constants includes the boolean values `true` and `false` and the unit value $\star$. The writing construct $loc := M$ rewrites the content of the location with the result of the evaluation of M. The calculus is simply-typed with recursive types (typing rules are as expected [Pie02]), and the type system ensures that the location has process type (the type of the embedded first-order processes). Hence, reading the content of the location always returns a process, and the term tested by $r?$ is a process.

The calculus is indeed an imperative λ-calculus with a one-place store, and the semantics and operators for interacting with the store of Λ^{loc} are those standard of such calculi. Extensions and variations of Λ^{loc}, and in particular the possibility of allowing more than one location in the calculus, are discussed in Section 3.6.1.

Reduction is defined on terms that are *closed* (i.e., without free variables) and *equipped with a location* containing a process P, i.e., configurations of the form $\langle P \, ; \, M \rangle$.

3.2.2 Nondeterministic processes

We consider both *call-by-name* and *call-by-value* reduction strategies. We call $\Lambda^{loc}_{\mathrm{N}}(\mathcal{L})$ the call-by-name language, $\Lambda^{loc}_{\mathrm{V}}(\mathcal{L})$ its call-by-value version, as usual omitting the parameter $\mathcal{L}$ when referring to the pure languages (without $\mathcal{L}$ processes). When $\mathcal{L}$ is an LTS, a reduction step has the form $\langle P \, ; \, M \rangle \longrightarrow \langle P' \, ; \, M' \rangle$, saying that the evaluation of M with P in the location produces a new term M' with process P' in the location. The rules for reduction are in Figure 3.2

In the call-by-name language $\Lambda^{loc}_{\mathrm{N}}(\mathcal{L})$, only the functional part of an application is evaluated, hence rule Beta-V and the production VC for evaluation contexts are omitted. In call-by-value, both the function and the argument of an application are evaluated, hence rule Beta-N is omitted in $\Lambda^{loc}_{\mathrm{V}}(\mathcal{L})$. In all these languages, although the operators of the λ-calculi themselves are sequential, nondeterministic computations are possible because the process in the location may present internal nondeterminism. As usual, $\Longrightarrow$ is the

$$\textsc{Beta-N}\ \frac{}{\langle P\,;\,(\lambda x.M_1)M_2\rangle \longrightarrow \langle P\,;\,M_1\{{}^{M_2}\!/\!_x\}\rangle}$$

$$\textsc{Beta-V}\ \frac{}{\langle P\,;\,(\lambda x.M)V\rangle \longrightarrow \langle P\,;\,M\{{}^{V}\!/\!_x\}\rangle}$$

$$\textsc{If1}\ \frac{}{\langle P\,;\,\texttt{if true then}\ M_1\ \texttt{else}\ M_2\rangle \longrightarrow \langle P\,;\,M_1\rangle}$$

$$\textsc{If2}\ \frac{}{\langle P\,;\,\texttt{if false then}\ M_1\ \texttt{else}\ M_2\rangle \longrightarrow \langle P\,;\,M_2\rangle}$$

$$\textsc{Write}\ \frac{}{\langle P'\,;\,loc := P\rangle \longrightarrow \langle P\,;\,\star\rangle} \qquad \textsc{Read}\ \frac{}{\langle P\,;\,!loc\rangle \longrightarrow \langle P\,;\,P\rangle}$$

$$\textsc{Act}\ \frac{P \xrightarrow{r} P'\ (\text{in } \mathcal{L})}{\langle P\,;\,r?\rangle \longrightarrow \langle P'\,;\,\texttt{true}\rangle} \qquad \textsc{RefAct}\ \frac{P \overset{r}{\not\longrightarrow}\ (\text{in } \mathcal{L})}{\langle P\,;\,r?\rangle \longrightarrow \langle P\,;\,\texttt{false}\rangle}$$

$$\textsc{Seq}\ \frac{}{\langle P\,;\,\star\,\underline{\texttt{seq}}\,M\rangle \longrightarrow \langle P\,;\,M\rangle}$$

$$\textsc{EvCon}\ \frac{C \text{ is an evaluation context} \quad \langle P\,;\,M\rangle \longrightarrow \langle P'\,;\,M'\rangle}{\langle P\,;\,C[M]\rangle \longrightarrow \langle P'\,;\,C[M']\rangle}$$

Evaluation contexts:

- call-by-name $C := [\cdot] \mid \texttt{if}\ C\ \texttt{then}\ M_1\ \texttt{else}\ M_2 \mid C\,\underline{\texttt{seq}}\,M \mid loc := C \mid CM$
- call-by-value $C := [\cdot] \mid \texttt{if}\ C\ \texttt{then}\ M_1\ \texttt{else}\ M_2 \mid C\,\underline{\texttt{seq}}\,M \mid loc := C \mid CM \mid VC$

Values: $V ::= c \mid \lambda x.M \mid P$

Figure 3.2: Reduction rules of $\Lambda^{loc}(\mathcal{L})$ for $\mathcal{L}$ an LTS

reflexive and transitive closure of ⟶.

In the call-by-name calculus $\Lambda^{loc}_{\mathrm{N}}(\mathcal{L})$, during a computation an $\mathcal{L}$ process may be moved around, may be copied, and may be placed into the location. However, once placed into the location for evaluation, the process cannot be stopped *and* later re-evaluated. Indeed, in call-by-name, when the Read rule is used, a value is produced and therefore the whole computation terminates.

In call-by-value, by contrast, the Read rule may be used by the argument of a function, and then the process so obtained may be passed to the function; as a consequence, the process may later be evaluated. This gives us more sophisticated process tests than call-by-name. Example 3.3 shows how the terms in $\Lambda^{loc}_{\mathrm{N}}$ and $\Lambda^{loc}_{\mathrm{V}}$ can test the existence of decorated traces in the behavior of a process placed in the location, and how some tests are available only in call-by-value.

Example 3.3. Below, a test is encoded in $\Lambda^{loc}_{\mathrm{V}}$ as a thunked boolean expression $\lambda.M$ and $(\lambda.M)\star$ is its 'unthunking'. (Thunking is useful in composition of tests. We assume here that thunked variables are all of unit type.) A test $\lambda.M$ is successful on a process P if there is a run starting from $\langle P\,;\,\lambda.M\rangle$ in which `true` is produced, i.e., $\langle P\,;\,(\lambda.M)\star\rangle \Longrightarrow$

$\langle P' \,;\, \texttt{true}\rangle$ for some P'.

$$\begin{aligned}
\mathrm{T}_a &\overset{\texttt{def}}{=} \lambda.a?\\
\mathrm{T}_{\neg a} &\overset{\texttt{def}}{=} \lambda.\,\texttt{if}\ \ a?\ \texttt{then false else true}\\
\underline{\mathrm{Seq}} &\overset{\texttt{def}}{=} \lambda x.\lambda y.\lambda.\,\texttt{if}\ \ x\star\ \ \texttt{then}\ \ y\star\ \ \texttt{else false}\\
\underline{\mathrm{And}} &\overset{\texttt{def}}{=} \lambda x.\lambda y.\lambda.((\lambda z.\,\texttt{if}\ \ x\star\ \ \texttt{then}\ \ (\mathit{loc} := z)\,\underline{\texttt{seq}}\,(y\star)\ \ \texttt{else false})!\mathit{loc})
\end{aligned}$$

Test T_a checks whether the process P in the location can perform action a (i.e., $P \xrightarrow{a}$). Dually, $\mathrm{T}_{\neg a}$ checks whether P is unable to perform a. Function $\underline{\mathrm{Seq}}$ composes the two argument tests sequentially. Thus, we can define the following test

$$M_1 \overset{\texttt{def}}{=} \underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\,\underline{\mathrm{Seq}}\ \mathrm{T}_{\neg c}\ \mathrm{T}_b\,)$$

that checks the existence of an a-derivative of the process in the location that cannot perform c but can perform b (i.e., $P \xrightarrow{a} P'$ with $P' \overset{c}{\not\longrightarrow}$ and $P' \xrightarrow{b}$, for some P'). Function $\underline{\mathrm{And}}$ makes the conjunction of the two argument tests. In general, for any pair M, N of tests, $\underline{\mathrm{And}}\,MN$ checks whether P passes both the test M and the test N. Thus the following term

$$M_2 \overset{\texttt{def}}{=} \underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\,\underline{\mathrm{And}}\ \mathrm{T}_b\ \mathrm{T}_c\,)$$

checks the existence of an a-derivative that can perform both b and c. Finally, term

$$M_3 \overset{\texttt{def}}{=} \underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\,\underline{\mathrm{And}}\,(\,\underline{\mathrm{Seq}}\ \mathrm{T}_b\ \mathrm{T}_c\,)(\,\underline{\mathrm{Seq}}\ \mathrm{T}_b\ \mathrm{T}_{\neg c}\,))$$

checks the existence of an a-derivative with both a b-derivative that can perform c and a b-derivative that cannot perform c.

In the call-by-name calculus $\Lambda^{\mathit{loc}}_{\mathrm{N}}$, while $\underline{\mathrm{Seq}}$ and T_a, $\mathrm{T}_{\neg a}$, M_1 have the same outcomes as in the call-by-value calculus $\Lambda^{\mathit{loc}}_{\mathrm{V}}$, function $\underline{\mathrm{And}}$ (and so also M_2, M_3) cannot be encoded. As a consequence, only the call-by-value calculi can separate

$$P \overset{\texttt{def}}{=} a.b + a.c \qquad\text{and}\qquad P' \overset{\texttt{def}}{=} a.(b+c) + P$$

When applied to P', test M_2 consumes an action a and then, in case the first a-branch of P' is taken, the whole expression reduces to

$$\langle b + c\,;\,(\lambda z.\,\texttt{if}\ \ \mathrm{T}_b\,\star\ \ \texttt{then}\ \ (\mathit{loc} := z)\,\underline{\texttt{seq}}\,(\,\mathrm{T}_c\star)\ \ \texttt{else false})!\mathit{loc}\rangle$$

Now, term $!\mathit{loc}$ is not a value, hence in call-by-value it is evaluated and produces process $b+c$. Since processes are values, $b+c$ is substituted for the variable z. Thus $b+c$ is placed in the location with which the test T_c is performed on the same process $b+c$, once the test T_b reports success. By contrast, in call-by-name $!\mathit{loc}$ is substituted for z before being evaluated, hence $b+c$ is lost before performing test T_c.

In λ-calculi, well-typed terms are supposed to produce computations that never get stuck. To maintain this property, we have to ensure that the action-test construct $a?$ returns a value even when the process in the location is unable to perform the requested action a. That is, we are allowed to observe the inability for the process to perform a certain action, in concurrency referred to as *action refusal* and usually omitted. We therefore consider also variants of the above λ-calculi without the refusal capability. Formally, rule RefAct is omitted. Of course, the price to pay is that computations from a well-typed term may get stuck. The call-by-name calculus without RefAct is called $\Lambda^{loc}_{\mathrm{N-ref}}(\mathcal{L})$, whereas call-by-value without RefAct is $\Lambda^{loc}_{\mathrm{V-ref}}(\mathcal{L})$.

Example 3.4. (We reuse P', M_1, M_3 from Example 3.3.) The processes P' and $P'' \overset{\mathtt{def}}{=} a.(b+c)$ are distinguished in $\Lambda^{loc}_{\mathrm{N}}$ and $\Lambda^{loc}_{\mathrm{V}}$ (via the test M_1), but they are equivalent in $\Lambda^{loc}_{\mathrm{N-ref}}$ and $\Lambda^{loc}_{\mathrm{V-ref}}$. In contrast, only in $\Lambda^{loc}_{\mathrm{V}}$ the processes

$$Q \overset{\mathtt{def}}{=} a.b.c + a.b \quad \text{and} \quad Q' \overset{\mathtt{def}}{=} a.(b.c + b)$$

can be separated via test M_3.

Theorem 3.6 summarizes the results on the discriminating power of the four λ-calculi when the embedded processes are fully nondeterministic.

Definition 3.1 of contextual equivalence is adapted to the λ-calculi supposing that the reduction relation $\longrightarrow$ is labeled with τ and adding the rule:

$$\textsc{Omega}\ \frac{}{\langle P\,;\,\mathtt{true}\rangle \xrightarrow{\omega}}$$

Contexts are terms with a single occurrence of a hole in one of the four languages defined above. The contexts contain no $\mathcal{L}$ processes, whilst the hole has process type.

In λ-calculi with store, the definition of contextual equivalence usually quantifies over all possible stores containing the locations occurring in the terms. In our case, this corresponds to quantifying over all possible assignments of the location to a process in $\mathcal{L}$ (for simplicity, we omit the case in which the location does not occur in the terms, and thereby the store would be empty). Hence, in this setting we have that processes P and Q in $\mathcal{L}$ are contextually equivalent in $\mathtt{AL}$ if for any unary context C of $\mathtt{AL}$ and for any process $P' \in \mathcal{L}$, $\langle P'\,;\,C[P]\rangle$ has a successful computation if and only if $\langle P'\,;\,C[Q]\rangle$ has a successful computation. In what follows, we use P_I to denote the process used to initialize the location in the definition of contextual equivalence.

Remark 3.5. The same results could be obtained by adding a constant c of process type to the syntax of Λ^{loc}, and then defining "M has a successful computation" as "$\langle c\,;\,M\rangle$ has a successful computation" (or, equivalently, by defining contexts as pairs of the form $\langle c\,;\,C\rangle$, where C is defined as above). This would yield a definition of contextual equivalence that immediately fits Definition 3.2, since it does not require the further quantification on all possible instantiations of the location. We choose the definition with the universal quantification over processes for continuity with the definitions in the literature of contextual equivalence on λ-calculi with store.

Theorem 3.6. *If $\mathcal{L}$ is an image-finite LTS, then:*

1. $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}} = \sim_{\mathrm{RS}}$ *(ready simulation equivalence);*

2. $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N}}} = \sim_{\mathrm{FTr}}$ *(failure trace equivalence);*

3. $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V-ref}}} = \sim_{\mathrm{S}}$ *(simulation equivalence);*

4. $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N-ref}}} = \sim_{\mathrm{Tr}}$ *(trace equivalence).*

Sketch. The proofs are different for inductive and coinductive equivalences. For the inductive equivalences, we discuss failure-trace equivalence and $\Lambda^{loc}_{\mathrm{N}}$ (item (2)). In one direction, one shows that for every failure trace ϕ there is a context C_ϕ of $\Lambda^{loc}_{\mathrm{N}}$ such that P has the failure trace ϕ iff $\langle P_I\,;\,C_\phi[P]\rangle$ produces `true`, for any process P_I initializing the location.

For the opposite direction, suppose P and Q have the same failure traces, and suppose $\langle P_I\,;\,C[P]\rangle \Longrightarrow\xrightarrow{\omega}$. We show that there is also a computation $\langle P_I\,;\,C[Q]\rangle \Longrightarrow\xrightarrow{\omega}$, proceeding as follows. Consider the computation $\langle P_I\,;\,C[P]\rangle \Longrightarrow\xrightarrow{\omega}$. First, we show that the reading capability in a boolean term in call-by-name is always followed by re-writing the location, and therefore has no impact on the computation. Then, to analyze what happens to the context and the processes inside them during the computation, we adopt an annotated operational semantics (equivalent to the original one) thanks to which we keep track of all the times a process is written in the location and of the observations made on the processes in the location (the transitions performed, the actions refused). Such observations are failure traces. Since process Q has the same failure traces as P, the configuration $\langle P_I\,;\,C[Q]\rangle$ can mimic the successful computation $\langle P_I\,;\,C[P]\rangle \Longrightarrow\xrightarrow{\omega}$. Note that during this computation the hole may get duplicated, and therefore the context may become polyadic.

As an example for the coinductive equivalences, we consider ready simulation equivalence and $\Lambda^{loc}_{\mathrm{V}}(\mathcal{L})$. In one direction, suppose P is not ready simulated by Q. By proposition 2.8, there is a formula F of Ready Simulation Logic that discriminates the two processes. We show that there is an encoding of these formulas to $\Lambda^{loc}_{\mathrm{V}}$-contexts, i.e., P satisfies F if and only if P in the context encoding F succeeds. Hence, there is a context discriminating the processes.

For the opposite direction, one shows that the relation

$$\mathcal{R} \overset{\mathsf{def}}{=} \{(\langle P\,;\,C[\widetilde{P}]\rangle, \langle Q\,;\,C[\widetilde{Q}]\rangle) \;\mid\; P, \widetilde{P} \text{ are pairwise ready simulated by } Q, \widetilde{Q}\}$$

where C is a polyadic context of $\Lambda^{loc}_{\mathrm{V}}$, is a strong ready simulation on reductions (in the sense that if $\langle P\,;\,C[\widetilde{P}]\rangle \;\mathcal{R}\; \langle Q\,;\,C[\widetilde{Q}]\rangle$ and $\langle P\,;\,C[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,M\rangle$ then $\langle P'\,;\,M\rangle = \langle P'\,;\,C[\widetilde{P'}]\rangle$ and $\langle Q\,;\,C[\widetilde{Q}]\rangle \longrightarrow \langle Q'\,;\,C[\widetilde{Q'}]\rangle$ with $\langle P'\,;\,C[\widetilde{P'}]\rangle\mathcal{R}\langle Q'\,;\,C[\widetilde{Q'}]\rangle$). As a consequence, any successful computation from $\langle P_I\,;\,C[P]\rangle$ may be mimicked by $\langle P_I\,;\,C[Q]\rangle$. More detailed proofs are presented in Section 3.7. □

3.2.3 Reactive probabilistic processes

The reduction relation for $\Lambda^{loc}(\mathcal{L})$ when $\mathcal{L}$ is an RPLTS is the expected probabilistic modification of the system for the nondeterministic case and is defined in Figure 3.3.

For any probability distribution Δ_1 on processes and for any distribution Δ_2 on terms of the language, we define the probability distribution $\langle \Delta_1\,;\,\Delta_2\rangle$ on configurations $\langle P\,;\,M\rangle$ as follows:

$$\langle \Delta_1\,;\,\Delta_2\rangle(\langle P\,;\,M\rangle) = \Delta_1(P)\cdot\Delta_2(M)$$

This is used to propagate the probability distribution reached from a process in the location to a λ-term, in rules Act and EvCon in Figure 3.3. Moreover, in rule EvCon, for a distribution Δ on terms and an evaluation context C we define the distribution on terms

$$\textsc{Beta-N}\ \frac{}{\langle P\,;\,(\lambda x.M_1)M_2\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,M_1\{{}^{M_2}\!/\!_{x}\}\rangle)}$$

$$\textsc{Beta-V}\ \frac{}{\langle P\,;\,(\lambda x.M)V\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,M\{{}^{V}\!/\!_{x}\}\rangle)}$$

$$\textsc{If1}\ \frac{}{\langle P\,;\,\mathtt{if\ true\ then}\ M_1\ \mathtt{else}\ M_2\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,M_1\rangle)}$$

$$\textsc{If2}\ \frac{}{\langle P\,;\,\mathtt{if\ false\ then}\ M_1\ \mathtt{else}\ M_2\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,M_2\rangle)}$$

$$\textsc{Write}\ \frac{}{\langle P'\,;\,loc := P\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,\star\rangle)} \qquad \textsc{Read}\ \frac{}{\langle P\,;\,!loc\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,P\rangle)}$$

$$\textsc{Act}\ \frac{P \xrightarrow{r} \Delta \text{ (in } \mathcal{L})}{\langle P\,;\,r?\rangle \longrightarrow \langle \Delta\,;\,\mathtt{dirac}(\mathtt{true})\rangle} \qquad \textsc{RefAct}\ \frac{P \stackrel{r}{\not\longrightarrow} \text{ (in } \mathcal{L})}{\langle P\,;\,r?\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,\mathtt{false}\rangle)}$$

$$\textsc{Seq}\ \frac{}{\langle P\,;\,\star\,\underline{\mathtt{seq}}\,M\rangle \longrightarrow \mathtt{dirac}(\langle P\,;\,M\rangle)}$$

$$\textsc{EvCon}\ \frac{C \text{ is an evaluation context} \quad \langle P\,;\,M\rangle \longrightarrow \langle \Delta_1\,;\,\Delta_2\rangle}{\langle P\,;\,C[M]\rangle \longrightarrow \langle \Delta_1\,;\,C[\Delta_2]\rangle}$$

Figure 3.3: Reduction rules of $\Lambda^{loc}(\mathcal{L})$ for $\mathcal{L}$ an RPLTS

$C[\Delta]$ as follows:

$$C[\Delta](M) = \begin{cases} \Delta(M') & \text{if } M = C[M'] \\ 0 & \text{otherwise} \end{cases}$$

The definitions of values and of evaluation contexts for call-by-name and call-by-value are the same as in the nondeterministic case (Figure 3.2).

Since the tested processes do not feature internal nondeterminism, all terms of $\Lambda^{loc}_{\mathrm{N}}$, $\Lambda^{loc}_{\mathrm{N-ref}}$, $\Lambda^{loc}_{\mathrm{V}}$ and $\Lambda^{loc}_{\mathrm{V-ref}}$ are reactive probabilistic processes, i.e., for any $P, \widetilde{P}$ and for any context C the operational semantics of $\langle P\,;\,C[\widetilde{P}]\rangle$ describes an RPLTS where states are configurations consisting of a term with a process in the location, and the transition are given by the reductions from such configurations to distributions over such configurations. Hence, for any context C of these languages and for any reactive probabilistic process P, $\langle P_I\,;\,C[P]\rangle$ has only one τ-labeled, maximal resolution on reductions.
Then we omit any reference to the resolutions and we write $\mathit{prob}(\mathcal{SC}(\langle P_I\,;\,C[P]\rangle))$ to denote $\mathit{prob}(\mathcal{SC}(z_{\langle P_I\,;\,C[P]\rangle}))$, where $z_{\langle P_I\,;\,C[P]\rangle}$ is the state associated with $\langle P_I\,;\,C[P]\rangle$ in the (unique) resolution.

Example 3.7. Consider the processes and λ-terms defined below, where $\underline{\mathrm{Seq}}$, $\underline{\mathrm{And}}$, T_a, $\mathrm{T}_{\neg a}$ are as in Example 3.3. In all four λ-calculi, the test M_1 separates between P and P'', since the resulting success probabilities are 0.5 and 0.25, respectively. The term M_2 distinguishes P and P' both in $\Lambda^{loc}_{\mathrm{N}}$ and in $\Lambda^{loc}_{\mathrm{V}}$, since $\mathit{prob}(\mathcal{SC}(\langle P\,;\,M_2\star\rangle)) = 0.5$ and $\mathit{prob}(\mathcal{SC}(\langle P'\,;\,M_2\star\rangle)) = 0$. The same processes are distinguished by M_3 in $\Lambda^{loc}_{\mathrm{V-ref}}$, which shows that the refusal operator is not necessary if the tested processes can be copied. Finally, only in $\Lambda^{loc}_{\mathrm{V-ref}}$ and $\Lambda^{loc}_{\mathrm{V}}$ the test M_4 distinguishes Q and Q', since it is only in call-by-value calculi that we can encode test $\underline{\mathrm{And}}$.

$$\begin{aligned}
P &\overset{\mathsf{def}}{=} a.((b + c.d) +_{0.5} (f + c.e))\\
P' &\overset{\mathsf{def}}{=} a.((b + c.e) +_{0.5} (f + c.d))\\
P'' &\overset{\mathsf{def}}{=} a.(((b + c.d) +_{0.5} c.d) +_{0.5} (f + c.e))\\
Q &\overset{\mathsf{def}}{=} a.(b.c +_{0.5} b)\\
Q' &\overset{\mathsf{def}}{=} a.b.(c +_{0.5} \mathbf{0})\\
M_1 &\overset{\mathsf{def}}{=} \underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\,\mathrm{T}_b\,)\\
M_2 &\overset{\mathsf{def}}{=} \underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\,\underline{\mathrm{Seq}}\ \mathrm{T}_{\neg f}\,(\,\underline{\mathrm{Seq}}\ \mathrm{T}_c\ \mathrm{T}_d\,))\\
M_3 &\overset{\mathsf{def}}{=} \underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\,\underline{\mathrm{And}}\ \mathrm{T}_b\,(\,\underline{\mathrm{Seq}}\ \mathrm{T}_c\ \mathrm{T}_d\,))\\
M_4 &\overset{\mathsf{def}}{=} \underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\,\underline{\mathrm{And}}\,(\,\underline{\mathrm{Seq}}\ \mathrm{T}_b\ \mathrm{T}_c\,)(\,\underline{\mathrm{Seq}}\ \mathrm{T}_b\ \mathrm{T}_c\,))
\end{aligned}$$

A peculiarity of probabilistic processes (with respect to nondeterministic processes) is that even when testing finite RPLTS there can be infinitely many successful computations from a term. This is because we are testing probabilistic systems using powerful languages such as λ-calculi, which can encode fixed-points. Hence, it does not generally hold that there exists a maximal length of the successful computations, as the following example shows.

Example 3.8. Let $P \overset{\mathsf{def}}{=} a.(b +_{0.5} \mathbf{0})$ and M be

$$\lambda y.(\lambda x.\,\underline{\mathrm{Seq}}\ \mathrm{T}_a\,(\lambda.\ \texttt{if}\ \ b?\ \texttt{then true else}\ \ (loc := x)\,\underline{\texttt{seq}}\,(yy\star)))!loc$$

In $\Lambda^{loc}_{\mathrm{V}}$, the term $\langle P\,;\, MM\star\rangle$ with probability 0.5 reports success and with probability 0.5 becomes again $\langle P\,;\, MM\star\rangle$. Thus, there are infinitely many successful computations from $\langle P\,;\, MM\star\rangle$, and the overall success probability is 1. (Note that the typing of M requires recursive types.)

Theorem 3.9. *If $\mathcal{L}$ is an RPLTS, then:*

1. $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}} = \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V-ref}}} = \sim_{\mathrm{PB}}$ *(probabilistic bisimilarity);*
2. $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N}}} = \sim_{\mathrm{PFTr}}$ *(probabilistic failure trace equivalence);*
3. $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N-ref}}} = \sim_{\mathrm{PTr}}$ *(probabilistic trace equivalence).*

Sketch. For the inductive equivalences, the proof schemata are as in the nondeterministic case, but we now have to reason directly at the level of probability distributions over configurations of the form $\langle P\,;\, M\rangle$. To this end, we redefine reductions and trace equivalences as relations over probability (sub)distributions, i.e., distributions that can have weight possibly smaller than one.

For item (1), in one direction we exploit the testing characterization of probabilistic bisimilarity presented in Section 2.3.3, using the language of tests $\mathbf{t} ::= \omega \;\big|\; r.\mathbf{t} \;\big|\; (\mathbf{t}_1, \mathbf{t}_2)$. We show that these tests are encodable in $\Lambda^{loc}_{\mathrm{V-ref}}$ (some hints are provided by Example 3.3 and Example 3.7), i.e., for every test $\mathbf{t}$ in $\mathbf{T}$ there is a term $M_{\mathbf{t}}$ such that for every P, $\Pr(\mathbf{t}, P) = \mathit{prob}(\mathcal{SC}(\langle P\,;\, M_{\mathbf{t}}\rangle))$. Hence, if $P \not\sim_{\mathrm{PB}} Q$ then there is a context of $\Lambda^{loc}_{\mathrm{V-ref}}$ that distinguishes P and Q. The same holds for the language $\Lambda^{loc}_{\mathrm{V}}$, since it includes $\Lambda^{loc}_{\mathrm{V-ref}}$.

For the other direction, the proof is in two steps, analogously to the case of nondeterministic processes: we first prove that if P, Q are probabilistic bisimilar $\mathcal{L}$ processes and C is a λ-calculus context, then also $\langle P_I \,;\, C[P]\rangle$ and $\langle P_I \,;\, C[Q]\rangle$ are probabilistic bisimilar when the bisimulation game is only played on reductions and success (ω) transitions. We then prove that, if $\langle P_I \,;\, C[P]\rangle$ and $\langle P_I \,;\, C[Q]\rangle$ are probabilistic bisimilar in this sense, then $\mathit{prob}(\mathcal{SC}(\langle P_I \,;\, C[P]\rangle)) = \mathit{prob}(\mathcal{SC}(\langle P_I \,;\, C[Q]\rangle))$, using the fact that probabilistic bisimilarity implies trace equivalence. More detailed proofs can be found in Section 3.7. □

3.3 Concurrency: syntax and operational rules

3.3.1 Syntax

We present here the concurrent languages used to test the first-order processes taken from an LTS or RPLTS $\mathcal{L}$. This section gives the syntax and operational rules. The following two sections study the equivalences induced by the languages. To simplify the presentation, we assume that also first-order communications exchange values, namely the unit value $\star$. *Names* include channels $a, b, \ldots$ and *locations* $l, m, \ldots$. The operators are those common to CCS and Higher-Order π-calculi. The special prefix ω indicates *success* of a computation. We add the basic constructs of calculi with passivation, namely the kell $[\![M]\!]_l$ and the passivation prefix $\mathtt{pass}_l(x).M$. The *refusal* prefix $\widetilde{r}_l.M$, where l is a location containing $\mathcal{L}$ processes, succeeds if the process in l is unable to perform the action r. (The addition of other operators is discussed in Section 3.6.3.) Kells may be nested, and the kell structure is transparent with respect to communications. In the remainder, unless otherwise stated, all mentioned processes are supposed to be *closed* (without free variables). A channel or prefix is *first-order* or *higher-order* depending on whether the exchanged value is $\star$ or is a process. We sometimes abbreviate first-order prefixes $a(x).M$ and $\overline{a}\langle\star\rangle.M$ as $a.M$ and $\overline{a}.M$ respectively, and omit the trailing $\mathbf{0}$ in $\alpha.\mathbf{0}$.

The language with all operators, $\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}$, is given in Figure 3.4. The subset without the refusal prefix is $\mathrm{HO}\pi_{\mathtt{pass}}$; the subset without passivation is $\mathrm{HO}\pi_{\mathrm{ref}}$; the subset without passivation and refusal is $\mathrm{HO}\pi$. These are the *higher-order* concurrent languages. In a *first-order* concurrent language, in contrast, all channels and prefixes are first-order (i.e., unit is the only communicable value) and the passivation prefix is disallowed. The resulting language is $\mathrm{CCS}^-_{\mathrm{ref}}$; when also refusal is disallowed, the language is CCS^-. (The '-' sign emphasizes the lack of the choice operator; see however Section 3.6.3.)

As usual, for any language, say `AL`, we write `AL`$(\mathcal{L})$ for the extension of `AL` with the first-order processes from the LTS or RPLTS $\mathcal{L}$, i.e., with the additional grammar production

$$M := \ldots \;\Big|\; P$$

where P is an $\mathcal{L}$ process. In the languages without passivation, the presence of kells is irrelevant; e.g., a process $N \mid [\![M]\!]_l$ behaves like $N \mid M$.

To avoid run-time errors in interactions, we assume a basic type system, that distinguishes three types of values: the unit value $\star$, the set $\mathit{Pr}_{\mathcal{L}}$ of tested $\mathcal{L}$ processes, and arbitrary processes $\mathit{Pr}_{\mathrm{all}}$, with the subtyping $\mathit{Pr}_{\mathcal{L}} \leq \mathit{Pr}_{\mathrm{all}}$. As a consequence, there are three types of names and variables. We distinguish the tested $\mathcal{L}$ processes from arbitrary processes because we allow refusal to act only on the former processes (this simplifies the operational rules, though it is not essential).

$$\begin{array}{rll}
u ::= a \;\mid\; l & & \text{(names)}\\
r ::= a \;\mid\; \overline{a} & & \text{(input/output channels)}\\
\alpha ::= a(x) \;\mid\; \overline{a}\langle M\rangle \;\mid\; \omega \;\mid\; \mathtt{pass}_l(x) \;\mid\; \widetilde{r}_l & & \text{(prefixes)}\\
M ::= M \mid M \;\mid\; \alpha.M \;\mid\; x \;\mid\; \mathbf{0} \;\mid\; [\![M]\!]_l \;\mid\; \star & & \text{(processes and values)}
\end{array}$$

Figure 3.4: Syntax for HO$\pi_{\mathtt{pass},\mathrm{ref}}$

$$\textsc{ParL}\;\frac{M_1 \xrightarrow{\mu} M_1'}{M_1 \mid M_2 \xrightarrow{\mu} M_1' \mid M_2} \qquad \textsc{ParR}\;\frac{M_2 \xrightarrow{\mu} M_2'}{M_1 \mid M_2 \xrightarrow{\mu} M_1 \mid M_2'}$$

$$\textsc{Com}\;\frac{M_1 \xrightarrow{\mu} M_1 \qquad M_2 \xrightarrow{\overline{\mu}} M_2'}{M_1 \mid M_2 \xrightarrow{\tau} M_1' \mid M_2'} \qquad \textsc{foPr}\;\frac{P \xrightarrow{r} P' \text{ (in } \mathcal{L})}{P \xrightarrow{r\langle\star\rangle} P'}$$

$$\textsc{Inp}\;\frac{}{a(x).M \xrightarrow{a\langle N\rangle} M\{{}^{N}\!/\!_{x}\}} \qquad \textsc{Out}\;\frac{}{\overline{a}\langle N\rangle.M \xrightarrow{\overline{a}\langle N\rangle} M}$$

$$\textsc{RefLoc}\;\frac{P \stackrel{r}{\not\longrightarrow} \text{ (in } \mathcal{L})}{[\![P]\!]_l \xrightarrow{\overline{\widetilde{r}_l}} [\![P]\!]_l} \qquad \textsc{RefPre}\;\frac{}{\widetilde{r}_l.N \xrightarrow{\widetilde{r}_l} N}$$

$$\textsc{Succ}\;\frac{}{\omega.M \xrightarrow{\omega} M} \qquad \textsc{Kell}\;\frac{M \xrightarrow{\mu} M'}{[\![M]\!]_l \xrightarrow{\mu} [\![M']\!]_l}$$

$$\textsc{PassLoc}\;\frac{}{[\![N]\!]_l \xrightarrow{\overline{\mathtt{pass}_l}N} \mathbf{0}} \qquad \textsc{PassPre}\;\frac{}{\mathtt{pass}_l(x).M \xrightarrow{\mathtt{pass}_l N} M\{{}^{N}\!/\!_{x}\}}$$

Figure 3.5: Operational semantics for HO$\pi_{\mathtt{pass},\mathrm{ref}}(\mathcal{L})$

3.3.2 Nondeterministic processes

The operational rules for the full language HO$\pi_{\mathtt{pass},\mathrm{ref}}(\mathcal{L})$, when $\mathcal{L}$ is an LTS, are presented in Figure 3.5. The grammar for action labels is:

$$\mu := \tau \;\Big|\; \overline{a}\langle M\rangle \;\Big|\; a\langle M\rangle \;\Big|\; \omega \;\Big|\; \overline{\mathtt{pass}_l}M \;\Big|\; \mathtt{pass}_l N \;\Big|\; \overline{\widetilde{r}_l} \;\Big|\; \widetilde{r}_l$$

where $\overline{a}\langle M\rangle$, $\overline{\mathtt{pass}_l}M$, and $\overline{\widetilde{r}_l}$ are the dual of, and synchronize with, $a\langle M\rangle$, $\mathtt{pass}_l N$, and $\widetilde{r}_l$. The dual of μ is $\overline{\mu}$. In the languages without refusal (HO$\pi_{\mathtt{pass}}(\mathcal{L})$, HO$\pi(\mathcal{L})$, CCS$^-(\mathcal{L})$), rules RefLoc and RefPre are missing; in the languages without passivation (HO$\pi_{\mathrm{ref}}(\mathcal{L})$, HO$\pi(\mathcal{L})$, CCS$^-(\mathcal{L})$, CCS$^-_{\mathrm{ref}}(\mathcal{L})$), rules PassLoc and PassPre are missing. Further, in the CCS languages the only value exchanged is $\star$.

3.3.3 Reactive probabilistic processes

If $\mathcal{L}$ is an RPLTS, the rules for parallel composition (Figure 3.6) propagate the probability distributions reached from the processes in $\mathcal{L}$. For any Δ_1, Δ_2, we define the distribution $\Delta_1 \mid \Delta_2$ on terms of the language as follows:

$$\Delta_1 \mid \Delta_2(M) \stackrel{\mathtt{def}}{=} \begin{cases} \Delta_1(M_1)\cdot\Delta_2(M_2) & \text{if } M = M_1 \mid M_2 \\ 0 & \text{otherwise} \end{cases}$$

$$\textsc{ParL}\ \frac{M_1 \xrightarrow{\mu} \Delta_1}{M_1 \mid M_2 \xrightarrow{\mu} \Delta_1 \mid \mathtt{dirac}(M_2)} \qquad \textsc{ParR}\ \frac{M_2 \xrightarrow{\mu} \Delta_2}{M_1 \mid M_2 \xrightarrow{\mu} \mathtt{dirac}(M_1) \mid \Delta_2}$$

$$\textsc{Com}\ \frac{M_1 \xrightarrow{\mu} \Delta_1 \qquad M_2 \xrightarrow{\overline{\mu}} \Delta_2}{M_1 \mid M_2 \xrightarrow{\tau} \Delta_1 \mid \Delta_2}$$

Figure 3.6: Rules for parallel composition in the probabilistic setting

Differently from the contexts of λ-calculi, now the contexts are nondeterministic. Therefore, in general $C[P]$ is a nondeterministic *and* probabilistic term (i.e., an NPLTS).

3.4 CCS languages: separation results

In the results on the concurrent languages, we include, in parenthesis, reference to the results for the λ-calculi to ease the comparison.

3.4.1 Nondeterministic processes

When $\mathcal{L}$ is an LTS, $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-}$ coincides with ordinary may-testing equivalence [DH84] and hence with $\sim_{\mathrm{Tr}}$, because the canonical tests of [DH84] can be encoded without resorting to the choice operator. For a similar reason, $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-_{\mathrm{ref}}}$ coincides with the refusal testing equivalence of [Phi87] and hence with $\sim_{\mathrm{FTr}}$.

3.4.2 Reactive probabilistic processes

When $\mathcal{L}$ is an RPLTS, the contextual equivalences induced by CCS^- and $\mathrm{CCS}^-_{\mathrm{ref}}$ are comprised between $\sim_{\mathrm{PB}}$ and $\sim_{\mathrm{PFTr}}$.

Theorem 3.10. *If $\mathcal{L}$ is an RPLTS, then:*

$$\sim_{\mathrm{PB}} \subsetneq \simeq^{\mathcal{L}}_{CCS^-_{\mathrm{ref}}} \subsetneq \simeq^{\mathcal{L}}_{CCS^-} \subsetneq \sim_{\mathrm{PFTr}} \ \ (= \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N}}})$$

Sketch. To prove the first inclusion, we exploit the congruence of $\sim_{\mathrm{PB}}$, and we observe that if $C[P_1]$ and $C[P_2]$ are bisimilar with respect to actions τ and ω then for every resolution of $C[P_1]$ there is a resolution of $C[P_2]$ that has the same probability of success, and vice-versa. The details of the proof can be derived from the proof of the more general result that $\sim_{\mathrm{PB}} \subseteq \simeq^{\mathcal{L}}_{\mathrm{HO}\pi}$, that we will present in the next section (Theorem 3.13, item (1)).

The second inclusion is immediate. To prove the third inclusion, we first show that contextual equivalence (even when only deterministic, non-probabilistic contexts are considered) implies probabilistic failure equivalence. Then we show how, for any process, the probability of an arbitrary failure trace ϕ can be recovered in terms of the probabilities of failures, from which the result follows. The details of the proof can be found in the proof of Theorem 4.3, presented in the following chapter.

The inclusion of $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-}$, and hence of $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-_{\mathrm{ref}}}$, in $\sim_{\mathrm{PFTr}}$ is strict: the $\sim_{\mathrm{PFTr}}$-equivalent processes

$$P \overset{\mathsf{def}}{=} a.((b.d + c.e) +_{0.5} (b.f + c.g))$$
$$P' \overset{\mathsf{def}}{=} a.(b.(d +_{0.5} f) + c.(e +_{0.5} g))$$

are distinguished by the CCS$^-$ context

$$C \overset{\mathsf{def}}{=} [\cdot] \mid \overline{a}.(\overline{b}.\overline{d}.\omega \mid \overline{c}.\overline{g}.\omega)$$

The maximum probability of succeeding for $C[P]$ is 1, whereas that of $C[P']$ is 0.5. The inclusion of $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-_{\mathrm{ref}}}$ in $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-}$ is strict as well: the processes Q, Q' in Example 3.7 are not distinguished in CCS$^-$, but they are distinguished by the CCS$^-_{\mathrm{ref}}$ context

$$C' \overset{\mathsf{def}}{=} [\![[\cdot]]\!]_l \mid \overline{a}.(\overline{b}.\overline{c}.\omega \mid \overline{b}.\widetilde{c}_l.\omega)$$

The maximum probabilities of success are respectively 1 and 0.5.

Finally, probabilistic bisimilarity is strictly included in $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-_{\mathrm{ref}}}$, since the non probabilistic bisimilar processes

$$R \overset{\mathsf{def}}{=} d.(e.Q +_{0.5} e.Q')$$
$$R' \overset{\mathsf{def}}{=} d.e.(Q +_{0.5} Q')$$

cannot be distinguished by any CCS$^-_{\mathrm{ref}}$-context. The different timing of the initial choices in R and R' is visible under bisimilarity but is not under the may semantics. (Intuitively, in the may semantics Q and Q' can only be distinguished by tests that exhibit success probabilities 1 and 0.5, respectively; we would need however a richer range of success probabilities to be able to separate R and R'.)[3] □

3.5 HOπ languages: separation results

3.5.1 Nondeterministic processes

The proof schemata for Theorem 3.11 are as those for the analogous results in λ-calculi; in one direction we essentially encode the (higher-order) separating tests of the λ-calculi into the Higher-Order π-calculi.

Theorem 3.11. *If $\mathcal{L}$ is an image-finite LTS, then:*

1. $\simeq^{\mathcal{L}}_{HO\pi_{\mathtt{pass},\mathrm{ref}}} = \sim_{\mathrm{RS}}$ *($= \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}}$);*
2. $\simeq^{\mathcal{L}}_{HO\pi_{\mathrm{ref}}} = \sim_{\mathrm{FTr}}$ *($= \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N}}}$);*
3. $\simeq^{\mathcal{L}}_{HO\pi_{\mathtt{pass}}} = \sim_{\mathrm{S}}$ *($= \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V-ref}}}$);*
4. $\simeq^{\mathcal{L}}_{HO\pi} = \sim_{\mathrm{Tr}}$ *($= \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N-ref}}}$).*

[3] We will further discuss analogous examples in the next chapter.

Example 3.12. In this example, we test a process P by placing it in a kell and then running a test M in parallel, as in $[\![P]\!]_l \mid M$. The tests $\overline{a}.\omega$ and $\widetilde{a}_l.\omega$ check whether $P \xrightarrow{a}$ and $P \overset{a}{\not\longrightarrow}$, respectively. The test M_1', below, in HOπ_{ref} performs the same test as the term M_1 in $\Lambda_{\mathrm{N}}^{loc}$ discussed in Example 3.3, while M_2' in HO$\pi_{\mathtt{pass}}$ corresponds to M_2 in $\Lambda_{\mathrm{V-ref}}^{loc}$ (the second occurrence $\mathtt{pass}_l(y)$ of the passivation operator destroys the first copy of the tested process, so as to ensure that the test $\overline{c}.\omega$ is executed on the second copy).

$$M_1' \stackrel{\mathtt{def}}{=} \overline{a}.\widetilde{c}_l.\overline{b}.\omega$$
$$M_2' \stackrel{\mathtt{def}}{=} \overline{a}.\mathtt{pass}_l(x).([\![x]\!]_l \mid \overline{b}.\mathtt{pass}_l(y).([\![x]\!]_l \mid \overline{c}.\omega))$$

3.5.2 Reactive probabilistic processes

The proof of Theorem 3.13(1) is analogous to the proofs for the λ-calculi $\Lambda_{\mathrm{V}}^{loc}$ and $\Lambda_{\mathrm{V-ref}}^{loc}$. Again, in one direction we essentially encode the separating tests of the λ-calculi. For the opposite direction, we show that bisimilarity on RPLTSs P_1 and P_2 implies bisimilarity on $C[P_1]$ and $C[P_2]$ with respect to actions τ and ω. Unlike in λ-calculi, we now have that, as in CCS$^-$, processes $C[P_1]$ and $C[P_2]$ are NPLTSs, and thereby have internal nondeterminism. Hence, we may have several resolutions (possibly an infinite number of resolutions). We show that for every resolution of $C[P_1]$ there is a resolution of $C[P_2]$ that has the same probability of success, and vice-versa.

Theorem 3.13. *If $\mathcal{L}$ is an RPLTS, then:*

1. $\simeq^{\mathcal{L}}_{HO\pi_{\mathtt{pass},\mathrm{ref}}} = \simeq^{\mathcal{L}}_{HO\pi_{\mathtt{pass}}} = \sim_{\mathrm{PB}}$ *(*$= \simeq^{\mathcal{L}}_{\Lambda_{\mathrm{V}}^{loc}} = \simeq^{\mathcal{L}}_{\Lambda_{\mathrm{V-ref}}^{loc}}$*);*
2. $\simeq^{\mathcal{L}}_{HO\pi_{\mathrm{ref}}} = \simeq^{\mathcal{L}}_{HO\pi} \subsetneq \simeq^{\mathcal{L}}_{CCS^-}$ *;*
3. $\simeq^{\mathcal{L}}_{HO\pi_{\mathrm{ref}}} \subsetneq \simeq^{\mathcal{L}}_{CCS^-_{\mathrm{ref}}}$.

The inclusions in (2) and (3) follow from the inclusions of the calculi. The following processes witness the strictness of the inclusion of $\simeq^{\mathcal{L}}_{\mathrm{HO}\pi}$ in $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-}$:

$$P \stackrel{\mathtt{def}}{=} d.Q + e.(f +_{0.5} \mathbf{0})$$
$$P' \stackrel{\mathtt{def}}{=} d.Q' + e.(f +_{0.5} \mathbf{0})$$

for Q, Q' as in Example 3.7. Processes P, P', different under $\sim_{\mathrm{PB}}$, are identified by $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-}$. They are also separated in HOπ, via the context

$$C \stackrel{\mathtt{def}}{=} \overline{h}\langle[\cdot]\rangle \mid h(x).(x \mid \overline{d}.\overline{a}.(\overline{b}.\overline{c}.\omega \mid \overline{b}.(x \mid \overline{e}) \mid \overline{f}.\omega))$$

Intuitively, this context uses higher-order communication to make copies of the tested process at the beginning, and then exploits the right-hand branch $e.(f +_{0.5} \mathbf{0})$ of the process itself in order to test the left-hand branch.

Analogously, let R, R' be as in Section 3.4.2 and define the processes

$$S \stackrel{\mathtt{def}}{=} R + f.(g +_{0.5} \mathbf{0}) + h.(i +_{0.6} \mathbf{0})$$
$$S' \stackrel{\mathtt{def}}{=} R' + f.(g +_{0.5} \mathbf{0}) + h.(i +_{0.6} \mathbf{0}).$$

It follows from $R \simeq^{\mathcal{L}}_{\mathrm{CCS}^-_{\mathrm{ref}}} R'$ that $S \simeq^{\mathcal{L}}_{\mathrm{CCS}^-_{\mathrm{ref}}} S'$, while the HO$\pi$-context C' distinguishes S and S'

$$C' \overset{\mathtt{def}}{=} \overline{j}\langle[\cdot]\rangle \mid j(x).(x \mid \overline{d}.(\overline{e}.\overline{a}.T \mid \overline{e}.(x \mid \overline{h}.\overline{i}.\omega)))$$
$$T \overset{\mathtt{def}}{=} \overline{b}.\overline{c}.\omega \mid \overline{b}.(x \mid \overline{f}.\overline{g}.\omega).$$

Since HO$\pi \subseteq$ HOπ_{ref}, the example shows the strictness of the inclusion of $\simeq^{\mathcal{L}}_{\mathrm{HO}\pi_{\mathrm{ref}}}$ in $\simeq^{\mathcal{L}}_{\mathrm{CCS}^-_{\mathrm{ref}}}$.

A summary of the results presented so far can be found in Chapter 5, figures 5.1 and 5.2.

3.6 Extensions and variations

3.6.1 Extending λ-calculi

Both in the call-by-name and in the call-by-value λ-calculi, a term is evaluated before being written in the store. Formally, this corresponds to the fact that the location can only be assigned to a process value (by rule Write only values can be written in the store), and $l := C$ is an evaluation context. This is a standard property of the semantics of functional languages with imperative feature, and it can be understood by noticing that if this were not the case then we could write in the store terms capable of performing operations over locations (e.g., reading, writing or testing a process, in our case). This, in turn, would require us to define another store with respect to which such terms should be evaluated.

The possibility of evaluating a term before assigning it to a location makes call-by-name computations closer to call-by-value ones. This can be seen by considering call-by-name calculi with a store that can contain more than one location, and with the operator for testing actions indexed by the location containing the process to be tested. The presence of multiple locations allows us to define the following term:

$$M \overset{\mathtt{def}}{=} \mathtt{if}\ a?_{l_1}\ \mathtt{then}\ (l_2 := !l_1)\ \underline{\mathtt{seq}}\ (l_3 := !l_1)\ \underline{\mathtt{seq}}\ N\ \mathtt{else\ false}$$

with $N \overset{\mathtt{def}}{=} \mathtt{if}\ b?_{l_2}\ \mathtt{then\ if}\ c?_{l_3}\ \mathtt{then\ true\ else\ false\ else\ false}$. If l_1 contains process P, term M allows us to check whether P can perform a and then reach a process performing both b and c. Since the evaluation of M is independent of the evaluation strategy, a call-by-name calculus with multiple locations could encode conjunctive tests and be as discriminating as the call-by-value calculus.

To prevent this, we have considered a call-by-name calculus with only one location. An alternative solution consists in defining a calculus with multiple locations which, however, does not allow the evaluation of a term before a location assignment.

By contrast, a store with multiple locations would not increase the discriminating power of call-by-value calculi.

3.6.2 Global vs. local communications

In our higher-order concurrent calculi, communications are *network transparent*, in the sense that they take place irrespective of the locations in which the interacting processes are placed. In the literature, this approach to communication is sometimes called *global*, as opposed to the *local* approach, where communication is subject to physical proximity. Under local communications, kells form communication barriers, because they confine where interactions can occur, with a finer control over communication interferences. This extra precision in communications, which can be obtained by formalizing local communications as in the Kell calculus [SS05], would not affect the discriminating power of the languages as far as contextual equivalences are concerned.

3.6.3 Other operators

The concurrent calculi we have considered do not include certain common operators, such as restriction, recursion, relabeling, and choice, so as to make the operational rules simpler or because often omitted in higher-order languages. The addition of these operators would not change the results presented (we assume that the hole of a context is not allowed to occur in recursive definitions). Some care is necessary with restriction in the presence of passivation, along the lines of [PS12a; KH13], because the lazy scope extrusion on restriction could allow contextual equivalence to make distinctions on processes solely on the basis of their free names [LSS11].

In HOπ we only allow communication of processes; the addition of process *abstractions*, or the name-passing communications of the π-calculus, as in the full HOπ, would not affect the results.

3.6.4 May vs. must equivalences

We have considered so far only contextual equivalences with success defined in the 'may' style. In the LTS case, the 'must' variants focus on the success of all maximal τ-computations (i.e., whose steps are all labeled with τ). With respect to Definition 3.1 the preorder $\leq^{\mathcal{L}}_{\mathtt{AL},\mathrm{must}}$ is introduced by requiring that, if all maximal τ-computations from $C[P_1]$ are successful, then so are those from $C[P_2]$. In the RPLTS case, the definition of $\leq^{\mathcal{L}}_{\mathtt{AL},\mathrm{must}}$ is obtained from Definition 3.2 by simply using $\sqcap$ in place of $\sqcup$, i.e., by considering the minimum probability of reaching success in the various maximal τ-resolutions.

The must-equivalences coincide with the may-equivalences when the tested processes are RPLTSs and the testing language is a λ-calculus, because internal nondeterminism does not occur ($C[P]$ is an RPLTS with a unique maximal resolution of nondeterminism).

In contrast, nondeterminism may spring up when the testing language is concurrent, or when the tested processes are LTSs rather than RPLTSs. We discuss below a few scenarios in which the relationship (and sometimes the coincidence) between must- and may-equivalences can be derived despite the presence of nondeterminism.

Let us start with LTSs. Due to the absence of divergence, in $\mathrm{CCS}^{-}_{\mathrm{ref}}$ the must-equivalence coincides with the refusal testing equivalence of [Phi87] and hence with $\sim_{\mathrm{FTr}}$; thus, it coincides with the may-equivalence. In contrast, it follows from [Nic87] that in CCS^{-} the must-equivalence coincides with $\sim_{\mathrm{F}}$ and hence is strictly finer than the may-equivalence, which is $\sim_{\mathrm{Tr}}$.

By contrast, we can show that, if the testing language is $\Lambda^{loc}_{\mathrm{V}}$ or $\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}$, then the reverse inclusion holds.

Theorem 3.14. *If $\mathcal{L}$ is an LTS, then $\leq^{\mathcal{L}}_{\mathtt{AL}} = \lesssim_{\mathrm{RS}} \subseteq \geq^{\mathcal{L}}_{\mathtt{AL},\mathrm{must}}$ for $\mathtt{AL} \in \{\Lambda^{loc}_{\mathrm{V}}, HO\pi_{\mathtt{pass},\mathrm{ref}}\}$.*

Proof. In the proof of Theorem 3.11, we have seen that if $P \lesssim_{\mathrm{RS}} Q$ then $C[P] \lesssim_{\mathrm{RS}} C[Q]$, with respect to actions τ and ω, for C a context of $\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}$. In order to show that $P \geq^{\mathcal{L}}_{\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}},\mathrm{must}} Q$, we prove that if all τ-labeled maximal computations from $C[Q]$ succeed then all τ-labeled maximal computations from $C[P]$ succeed. We show the result by contraposition. If $C[P]$ has a τ-labeled maximal computations that does not succeed, one of the following holds:

- $C[P]$ has a finite τ-labeled computation such that all states in the computation cannot perform ω, and the last state is stuck;
- $C[P]$ has an infinite τ-labeled computation such that all states in the computation cannot perform ω.

It is easy to see by induction on n that if $C[P]\,\mathcal{R}\,C[Q]$ for some ready simulation $\mathcal{R}$ with respect to actions τ and ω and $C[P] = M_0 \xrightarrow{\tau} M_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} M_n$, with $M_i \overset{\omega}{\not\longrightarrow}$ for $0 \leq i \leq n$, then $C[Q] = N_0 \xrightarrow{\tau} N_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} N_n$ with $N_i \overset{\omega}{\not\longrightarrow}$ and $M_i\,\mathcal{R}\,N_i'$ for $0 \leq i \leq n$. Hence, in both cases described above, process $C[Q]$ has a corresponding path that does not succeed, which implies the result.
The result for $\Lambda^{loc}_{\mathrm{V}}$ follows analogously, using the proof of Theorem 3.6 which implies that if $P \lesssim_{\mathrm{RS}} Q$ then $\langle P_I\,;\,C[P]\rangle \lesssim_{\mathrm{RS}} \langle P_I\,;\,C[Q]\rangle$ with respect to actions τ and ω, for C a context of $\Lambda^{loc}_{\mathrm{V}}$. □

As a corollary, we have that may-equivalence implies must-equivalence on LTSs for these languages.

Corollary 3.15. *If $\mathcal{L}$ is an LTS, then $\simeq^{\mathcal{L}}_{\mathtt{AL}} = \sim_{\mathrm{RS}} \subseteq \simeq^{\mathcal{L}}_{\mathtt{AL},\mathrm{must}}$ for $\mathtt{AL} \in \{\Lambda^{loc}_{\mathrm{V}}, HO\pi_{\mathtt{pass},\mathrm{ref}}\}$.*

The refusal operator (respectively, the RefAct rule in λ-calculus) is essential for the inclusion to hold: the processes $P \overset{\mathtt{def}}{=} a.b + a$ and $P' \overset{\mathtt{def}}{=} a.b$ are $\sim_{\mathrm{S}}$-equivalent and hence may-equivalent both in $\Lambda^{loc}_{\mathrm{V-ref}}$ and in $\mathrm{HO}\pi_{\mathtt{pass}}$, but only P' always succeeds when the trace $a\,b$ is tested.

We now move to RPLTSs. By Theorem 3.6, the tests needed in order to distinguish $\sim_{\mathrm{PB}}$-inequivalent RPLTSs are encodable in $\Lambda^{loc}_{\mathrm{V-ref}}$. The passivation operator allows us to encode these tests in $\mathrm{HO}\pi_{\mathtt{pass}}$ without losing the sequentiality of the tests. Since RPLTSs do not have internal nondeterminism and the tests are sequential, the resulting NPLTS has a unique maximal resolution. Hence, $\sim_{\mathrm{PB}}$-inequivalent RPLTSs are neither may-equivalent nor must-equivalent. Moreover, the same argument used to show that $\sim_{\mathrm{PB}}$ implies may-equivalence can be adapted to the must-case. We have seen in the proof of Theorem 3.13 that, on RPLTS, $P \sim_{\mathrm{PB}} Q$ implies $C[P] \sim_{\mathrm{PB}} C[Q]$ with respect to actions τ and ω, for C a context of $\mathrm{HO}\pi_{\mathtt{pass}}$ or $\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}$. We have also seen how this implies that for every resolution of $C[P]$ there is a resolution of $C[Q]$ such that $\mathit{prob}(\mathcal{SC}(z_{C[P]})) = \mathit{prob}(\mathcal{SC}(z_{C[Q]}))$, and vice versa for every resolution of $C[Q]$ there is a resolution of $C[P]$ such that $\mathit{prob}(\mathcal{SC}(z_{C[P]})) = \mathit{prob}(\mathcal{SC}(z_{C[Q]}))$. Hence, we can conclude that

$$\bigsqcap_{\mathcal{Z}_{C[P]} \in \mathit{Res}_{\tau,max}(C[P])} \mathit{prob}(\mathcal{SC}(z_{C[P]})) = \bigsqcap_{\mathcal{Z}_{C[Q]} \in \mathit{Res}_{\tau,max}(C[Q])} \mathit{prob}(\mathcal{SC}(z_{C[Q]}))$$

and thereby $\sim_{\mathrm{PB}} \subseteq \simeq^{\mathcal{L}}_{\mathtt{AL},\mathrm{must}}$.

Theorem 3.16. *If $\mathcal{L}$ is an RPLTS, then $\simeq^{\mathcal{L}}_{\mathtt{AL},\mathrm{must}} = \simeq^{\mathcal{L}}_{\mathtt{AL}} = \sim_{\mathrm{PB}}$, for $\mathtt{AL} \in \{HO\pi_{\mathtt{pass}}, HO\pi_{\mathtt{pass},\mathrm{ref}}\}$.*

3.7 Proofs

Proof of Theorem 3.6 - left-to-right inclusions

We prove the left-to-right inclusions of Theorem 3.6, by showing how to encode tests that discriminate non-equivalent (with respect to simulation-like or trace-like equivalences) processes.

For item (1) ($\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}} \subseteq \sim_{\mathrm{RS}}$), we prove that there is an encoding function $\mathrm{Enc}(\cdot)$ from the Ready Simulation Logic formulas (Section 2.2.4) to $\Lambda^{loc}_{\mathrm{V}}$-terms such that $P \models F$ if and only if $\langle P\,;\mathrm{Enc}(F)\rangle$ has a successful computation (i.e., $\langle P\,;\mathrm{Enc}(F)\rangle \Longrightarrow\xrightarrow{\omega}$). Hence, if P and Q are not ready simulation equivalent then (by Proposition 2.8) there is a formula F such that $P \models F$ and $Q \not\models F$, which is equivalent to saying that $\langle P\,;\mathrm{Enc}(F)\rangle \Longrightarrow\xrightarrow{\omega}$ and $\langle Q\,;\mathrm{Enc}(F)\rangle \not\Longrightarrow\xrightarrow{\omega}$. Then, by defining context $C_F = loc := [\cdot]\,\underline{\mathtt{seq}}\,\mathrm{Enc}(F)$ and noting that $\langle P_I\,; C_F[P']\rangle \Longrightarrow\xrightarrow{\omega}$ iff $\langle P'\,;\mathrm{Enc}(F)\rangle \Longrightarrow\xrightarrow{\omega}$ for any P', we derive that $\langle P_I\,; C_F[P]\rangle \Longrightarrow\xrightarrow{\omega}$ and $\langle P_I\,; C_F[Q]\rangle \Longrightarrow\xrightarrow{\omega}$. Therefore, $P \not\sim_{\mathrm{RS}} Q$ implies $P \not\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}} Q$.

We define the encoding as follows:

$$
\begin{aligned}
&\mathrm{Enc}(\top) = \mathtt{true}\\
&\mathrm{Enc}(\neg r) = \mathtt{if}\ r?\ \mathtt{then\ false\ else\ true}\\
&\mathrm{Enc}(\langle\, r\,\rangle F) = \mathtt{if}\ r?\ \mathtt{then}\ \mathrm{Enc}(F)\ \mathtt{else\ false}\\
&\mathrm{Enc}(F_1 \wedge F_2) = (\lambda z.\,\mathtt{if}\ \mathrm{Enc}(F_1)\ \mathtt{then}\ (loc := z)\,\underline{\mathtt{seq}}\,\mathrm{Enc}(F_2)\ \mathtt{else\ false})!loc
\end{aligned}
$$

We prove by structural induction on F that $P \models F$ if and only if $\langle P\,;\mathrm{Enc}(F)\rangle \Longrightarrow\xrightarrow{\omega}$. The cases $F = \top$ and $F = \neg r$ immediately follow from the operational semantics of $\Lambda^{loc}_{\mathrm{V}}$. If $F = \langle\, r\,\rangle F'$ then the statement follows from the inductive hypothesis: if $P \xrightarrow{r} P'$ for some P' such that $P' \models F'$ then $\langle P\,;\mathrm{Enc}(F)\rangle \Longrightarrow \langle P'\,;\mathrm{Enc}(F')\rangle \Longrightarrow\xrightarrow{\omega}$. Otherwise, either $P \overset{r}{\not\longrightarrow}$, in which case $\langle P\,;\mathrm{Enc}(F)\rangle \Longrightarrow \langle P\,;\mathtt{false}\rangle$, or $P \not\models F$ for all P' that P reaches by doing r. In this case, by the inductive hypothesis we have that for every $\langle P'\,; M\rangle$ such that $\langle P\,;\mathrm{Enc}(F)\rangle \longrightarrow \langle P'\,; M\rangle$, $\langle P'\,; M\rangle$ never performs ω.
Finally, suppose that $F = F_1 \wedge F_2$. If P satisfies the formula, then we have the following sequence of reductions:

$$
\begin{aligned}
&\langle P\,;\mathrm{Enc}(F_1 \wedge F_2)\rangle \Longrightarrow\\
&\langle P\,;\ \mathtt{if}\ \mathrm{Enc}(F_1)\ \mathtt{then}\ (loc := P)\,\underline{\mathtt{seq}}\,\mathrm{Enc}(F_2)\ \mathtt{else\ false}\rangle \Longrightarrow \quad \text{(inductive hypothesis)}\\
&\langle P'\,;\ \mathtt{if\ true\ then}\ (loc := P)\,\underline{\mathtt{seq}}\,\mathrm{Enc}(F_2)\ \mathtt{else\ false}\rangle \longrightarrow\\
&\langle P'\,; (loc := P)\,\underline{\mathtt{seq}}\,\mathrm{Enc}(F_2)\rangle \Longrightarrow\\
&\langle P\,;\mathrm{Enc}(F_2)\rangle \Longrightarrow \quad \text{(inductive hypothesis)}\\
&\langle P''\,;\mathtt{true}\rangle
\end{aligned}
$$

If F is false at P and $P \not\models F_1$ then

$$\langle P\,;\mathrm{Enc}(F_1 \wedge F_2)\rangle \Longrightarrow \langle P\,;\ \mathtt{if}\ \mathrm{Enc}(F_1)\ \mathtt{then}\ (loc := P)\,\underline{\mathtt{seq}}\,\mathrm{Enc}(F_2)\ \mathtt{else\ false}\rangle$$

and by the inductive hypothesis $\mathrm{Enc}(F_1)$ never becomes `true`. The case when $P \models F_1$ and $P \not\models F_2$ is analogous.

The same encoding allows us to prove that for every formula F of Simulation Logic, $P \models F$ if and only if $\langle P\,;\,\mathrm{Enc}(F)\rangle \Longrightarrow\xrightarrow{\omega}$. Since the encoding of Simulation Logic does not exploit the capability of observing action refusal, we can use $\Lambda^{loc}_{\mathrm{V-ref}}$ as target language. Thus, $P \not\sim_{\mathrm{S}} Q$ implies $P \not\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V-ref}}} Q$.

For failure trace equivalence, we prove that for any failure trace $\phi = (r_i, F_i)_{i\le n} \in (\mathcal{A} \times 2^{\mathcal{A}})^*$, process P has the failure trace ϕ if and if $\langle P\,;\,t_\phi\rangle \Longrightarrow\xrightarrow{\omega}$, where t_ϕ is defined by induction on n:

$$\begin{array}{l} t_\emptyset = \mathtt{true} \\ t_{(r_i,F_i)_{1\le i\le n+1}} = \mathtt{if}\ r_{n+1}?\ \mathtt{then}\ \mathtt{if}\ t_{F_{n+1}}\ \mathtt{then}\ t_{(r_i,F_i)_{1\le i\le n}}\ \mathtt{else\ false\ else\ false} \end{array}$$

and for any failure set F, term t_F is `true` if F is empty, while for $r \in F$ and $F' = F \setminus \{r\}$ we have $t_F = \mathtt{if}\ r?\ \mathtt{then\ false\ else}\ t_{F'}$ (we assume some ordering on the labels in the failure set F, so that the term t_F is uniquely determined by labels in the set).

Analogously, for traces we can build discriminating tests by considering the term $t_{(r_i,\emptyset)_{i\le n}}$, for any trace $r_1, ..., r_n$. In this case, rule RefAct is not necessary, hence test $t_{(r_i,\emptyset)_{i\le n}}$ can be built in $\Lambda^{loc}_{\mathrm{V-ref}}$ as well.

Proof of Theorem 3.6 - right-to-left inclusions

We start from the coinductive equivalences, and in particular from item (1) ($\simeq_{\mathrm{RS}} \subseteq \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}}$). We first prove that that the following relation is a ready simulation on reductions:

$$\mathcal{R} \overset{\mathsf{def}}{=} \{(\langle P\,;\,C[\widetilde{P}]\rangle, \langle Q\,;\,C[\widetilde{Q}]\rangle) \ \mid\ P, \widetilde{P} \lesssim_{\mathrm{RS}} Q, \widetilde{Q}\ \}$$

where C is a polyadic context of $\Lambda^{loc}_{\mathrm{V}}$.

Consider term $\langle P\,;\,C[\widetilde{P}]\rangle$ and let $P, \widetilde{P} \lesssim_{\mathrm{RS}} Q, \widetilde{Q}$. We prove by structural induction on C that if $\langle P\,;\,C[\widetilde{P}]\rangle \mathcal{R} \langle Q\,;\,C[\widetilde{Q}]\rangle$ and $\langle P\,;\,C[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,M\rangle$ then $\langle Q\,;\,C[\widetilde{Q}]\rangle \longrightarrow \langle Q'\,;\,N\rangle$ with $\langle P'\,;\,M\rangle \mathcal{R} \langle Q'\,;\,N\rangle$.

- If $C = [\cdot]$ or C is a constant or $C = \lambda x.C'$ then both terms cannot perform any reduction.

- If $C = C_1C_2$ then $C[\widetilde{P}] = C_1[\widetilde{P}]C_2[\widetilde{P}]$ and $C[\widetilde{Q}] = C_1[\widetilde{Q}]C_2[\widetilde{Q}]$ with $\widetilde{P} \lesssim_{\mathrm{RS}} \widetilde{Q}$. We have two cases. If $\langle P\,;\,C_1[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,M\rangle$ we have by the inductive hypothesis that $M = \langle P'\,;\,C_1'[\widetilde{P}']\rangle$ and $\langle Q\,;\,C_1[\widetilde{Q}]\rangle \longrightarrow \langle Q'\,;\,C_1'[\widetilde{Q}']\rangle$ with $P', \widetilde{P}' \lesssim_{\mathrm{RS}} Q', \widetilde{Q}'$. Hence,

  $$\begin{array}{l} \langle P\,;\,C_1[\widetilde{P}]C_2[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,C_1'[\widetilde{P}']C_2[\widetilde{P}]\rangle \\ \langle Q\,;\,C_1[\widetilde{Q}]C_2[\widetilde{Q}]\rangle \longrightarrow \langle Q'\,;\,C_1'[\widetilde{Q}']C_2[\widetilde{Q}]\rangle \end{array}$$

  and the reached configurations are related.

  If $C_1[\widetilde{P}]$ is a value then if $\langle P\,;\,C_2[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,C_2'[\widetilde{P}']\rangle$ we have the same reasoning.

  If both $C_1[\widetilde{P}]$ and $C_2[\widetilde{P}]$ are values then $C_1[\widetilde{P}] = \lambda x.C_3[\widetilde{P}]$ and

  $$\begin{array}{l} \langle P\,;\,C_1[\widetilde{P}]C_2[\widetilde{P}]\rangle \longrightarrow \langle P\,;\,C_3[\widetilde{P}]\{{}^{C_2[\widetilde{P}]}\!/\!{}_x\}\rangle \\ \langle Q\,;\,C_1[\widetilde{Q}]C_2[\widetilde{Q}]\rangle \longrightarrow \langle Q\,;\,C_3[\widetilde{Q}]\{{}^{C_2[\widetilde{Q}]}\!/\!{}_x\}\rangle \end{array}$$

Since x cannot occur in processes, $\langle P\,;\,C_3[\widetilde{P}]\{{}^{C_2[\widetilde{P}]}\!/\!_x\}\rangle = \langle P\,;\,C_3\{{}^{C_2}\!/\!_x\}[\widetilde{P}]\rangle$ is related to $\langle Q\,;\,C_3[\widetilde{Q}]\{{}^{C_2[\widetilde{Q}]}\!/\!_x\}\rangle = \langle Q\,;\,C_3\{{}^{C_2}\!/\!_x\}[\widetilde{Q}]\rangle$.

- If $C = \texttt{if}\ C_1\ \texttt{then}\ C_2\ \texttt{else}\ C_3$ and $\langle P\,;\,C[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,M\rangle$ then we have three cases. If C_1 is $\texttt{true}$ then $\langle P\,;\,C[\widetilde{P}]\rangle \longrightarrow \langle P\,;\,C_2[\widetilde{P}]\rangle$ and $\langle Q\,;\,C[\widetilde{Q}]\rangle \longrightarrow \langle Q\,;\,C_2[\widetilde{Q}]\rangle$ with $\langle P\,;\,C_2[\widetilde{P}]\rangle\,\mathcal{R}\,\langle Q\,;\,C_2[\widetilde{Q}]\rangle$.

  The case when $C_1 = \texttt{false}$ is symmetric.

  Otherwise, $\langle P\,;\,C_1[\widetilde{P}]\rangle \longrightarrow \langle P\,;\,C_1'[\widetilde{P'}]\rangle$ and rule EvCon is used to derive the transition. In this case, the conclusion follows from the inductive hypothesis.

- If $C = C_1\,\underline{\texttt{seq}}\,C_2$ and $C_1 = \star$ then by rule Seq both $\langle P\,;\,C[\widetilde{P}]\rangle$ and $\langle Q\,;\,C[\widetilde{Q}]\rangle$ reach configurations that are in relation $\mathcal{R}$. If $\langle P\,;\,C_1[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,M\rangle$ then the result follows from the inductive hypothesis.

- If $C = !loc$ we have $\langle P\,;\,!loc\rangle \longrightarrow \langle P\,;\,P\rangle$ and $\langle Q\,;\,!loc\rangle \longrightarrow \langle Q\,;\,Q\rangle$ and the result follows.

- If $C = loc := C_1$ then:

  - if $C_1[\widetilde{P}]$ is process P_1 then $\langle P\,;\,loc := P_1\rangle \longrightarrow \langle P_1\,;\,\star\rangle$ and $C_1[\widetilde{Q}] = Q_1$. Therefore, $\langle Q\,;\,loc := Q_1\rangle \longrightarrow \langle Q_1\,;\,\star\rangle$, with $P_1 \lesssim_{\mathrm{RS}} Q_1$;
  - if $\langle P\,;\,C_1[\widetilde{P}]\rangle \longrightarrow \langle P'\,;\,M\rangle$ then by the inductive hypothesis we have that $M = C_1'[\widetilde{P'}]$ and $\langle Q\,;\,C_1[\widetilde{Q}]\rangle \longrightarrow \langle Q'\,;\,C_1'[\widetilde{Q'}]\rangle$ and the configurations reached are in $\mathcal{R}$, hence the result follows.

- If $C = r?$ then we have two cases:

  - if $P \xrightarrow{r} P'$ then $\langle P\,;\,r?\rangle \longrightarrow \langle P'\,;\,\texttt{true}\rangle$ and, since P is ready simulated by Q, $Q \xrightarrow{r} Q'$ with $P' \lesssim_{\mathrm{RS}} Q'$ and $\langle Q\,;\,r?\rangle \longrightarrow \langle Q'\,;\,\texttt{true}\rangle$, and the configurations are related.
  - If $P \overset{r}{\not\rightarrow}$ then $Q \overset{r}{\not\rightarrow}$ and the configurations reached are $\langle P\,;\,\texttt{false}\rangle\mathcal{R}\langle Q\,;\,\texttt{false}\rangle$.

Hence, since trivially $P_I \lesssim_{\mathrm{RS}} P_I$, if there is a path $\langle P_I\,;\,C[P]\rangle \Longrightarrow \langle P'\,;\,\texttt{true}\rangle$ then there is a path $\langle P_I\,;\,C[Q]\rangle \Longrightarrow \langle Q'\,;\,\texttt{true}\rangle$ for some P', Q'. Therefore, $P \simeq_{\mathrm{RS}} Q$ implies $P {\simeq}^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}} Q$.

The proof for $P \sim_{\mathrm{S}} Q$ implies $P {\simeq}^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V-ref}}} Q$ is analogous, the only difference being that we use simulations instead of ready simulations.

We now prove the right-to-left inclusions for the inductive equivalences, starting from trace equivalence ($\sim_{\mathrm{Tr}} \subseteq {\simeq}^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N-ref}}}$). The proof is based on the fact that in call-by-name the presence of a successful computation is not affected by the reading capability, as we now show. We first note that the following holds:

Lemma 3.17. *$\langle P\,;\,M\rangle \longrightarrow \langle P'\,;\,N\rangle$ has a derivation where the* Read *axiom is used iff either $M = !loc$ or $M = E[loc := !loc]$ for E an evaluation context.*

Proof. The right-to-left implication is trivial, while the other direction follows by induction on the derivation of $\langle P\,;\,M\rangle \longrightarrow \langle P'\,;\,N\rangle$. We have two cases: either the derivation is the READ axiom, in which case $M = !loc$, or the derivation has been derived using the rule for evaluation contexts, i.e., there is an evaluation context E such that $M = E[M_1]$, $N = E[M_2]$ and $\langle P\,;\,M\rangle \longrightarrow \langle P'\,;\,N\rangle$ is derived by $\langle P\,;\,M_1\rangle \longrightarrow \langle P'\,;\,M_2\rangle$. Then by the inductive hypothesis on $\langle P\,;\,M_1\rangle \longrightarrow \langle P'\,;\,M_2\rangle$ we have two cases. Either $M_1 = E'[loc := !loc]$, and the result follows, or $M_1 = !loc$. In this case, we prove by induction on the definition of E that either $E = [\cdot]$ or $E = E'[loc := [\cdot]]$, from which the result follows.
If $E = [\cdot]$ then the property holds by definition. For the inductive cases, if $E = loc := E'$ then by the inductive hypothesis either $E' = E''[loc := [\cdot]]$ or $E' = [\cdot]$, and in both cases we have that E is of the form $E'''[loc := [\cdot]]$ for some E'''. Otherwise (if $E = E'\,\underline{\texttt{seq}}\,M$ or $E = E'M$ or $E = \texttt{if}\ E'\ \texttt{then}\ M\ \texttt{else}\ N$) then, by typing, E' cannot be of the form $[\cdot]$, so by the inductive hypothesis $E' = E''[loc := [\cdot]]$ and the property holds. □

Since in a derivation of the form $\langle P\,;\,M\rangle \Longrightarrow \langle P'\,;\,\texttt{true}\rangle$ the terms to which M reduces to cannot be equal to $!loc$, it follows from Lemma 3.17 that $\langle P\,;\,M\rangle \Longrightarrow \langle P'\,;\,\texttt{true}\rangle$ iff $\langle P\,;\,M\rangle \Longrightarrow^- \langle P'\,;\,\texttt{true}\rangle$, where $\Longrightarrow^-$ is defined as $\Longrightarrow$ except for the fact that we substitute all pairs of transitions of the form

$$\langle P\,;\,E'[loc := !loc]\rangle \longrightarrow \langle P\,;\,E'[loc := P]\rangle \longrightarrow \langle P\,;\,E'[\star]\rangle$$

with

$$\langle P\,;\,E'[loc := !loc]\rangle \longrightarrow \langle P\,;\,E'[\star]\rangle$$

Since these steps are deterministic and have no effect on the value in the location, in what follows we can assume without loss of generality to have an operational semantics where the READ rule is substituted by

$$\text{READ'}\ \frac{}{\langle P\,;\,loc := !loc\rangle \longrightarrow \langle P\,;\,\star\rangle}$$

In the remainder of the proof, we omit the symbol $^-$ and we directly use $\longrightarrow$ and $\Longrightarrow$ to denote $\longrightarrow^-$ and $\Longrightarrow^-$.

Based on this semantics, we now define the relation $\overset{\mu}{\longmapsto}$ as follows, for $\mu = W, r, \tau$:

- $\langle P\,;\,M\rangle \overset{W}{\longmapsto} \langle P'\,;\,M'\rangle$ if the derivation of the transition $\langle P\,;\,M\rangle \longrightarrow \langle P'\,;\,M'\rangle$ uses the WRITE rule;
- $\langle P\,;\,M\rangle \overset{r}{\longmapsto} \langle P'\,;\,M'\rangle$ if the derivation of the transition $\langle P\,;\,M\rangle \longrightarrow \langle P'\,;\,M'\rangle$ uses the ACT rule on label r;
- $\langle P\,;\,M\rangle \overset{\tau}{\longmapsto} \langle P'\,;\,M'\rangle$ otherwise.

This relation is well defined since the rules WRITE and ACT cannot be used both in a derivation, and they occur at most once in a derivation. For $\mu' = W, r$ and σ a sequence of labels in μ', we define the weak labeled transition $\langle P\,;\,M\rangle \overset{\mu'}{\Longmapsto} \langle P'\,;\,M'\rangle$ (where any finite number of τ-transitions can occur before and after action μ is performed) and its reflexive and transitive closure $\langle P\,;\,M\rangle \overset{\sigma}{\Longmapsto} \langle P'\,;\,M'\rangle$ as usual, for $\sigma \in (\{W\} \cup \mathcal{A})^*$

We prove the following lemmas. If not specified otherwise, contexts are assumed to be polyadic and process names do not occur in the contexts, i.e., they are pure contexts of the λ-calculus language. We say that C is a P-evaluation context if it is an evaluation context where the only process name that can occur is P. For $\sigma \in \mathcal{A}^*$, we write $\xrightarrow{\sigma}$ for the reflexive and transitive closure of $\xrightarrow{r}$ on processes, i.e., $P \xrightarrow{\sigma} P'$ if there is a computation labeled by σ from P to P'.

Lemma 3.18. *If* $\langle P' \,;\, C[P]\rangle \Longmapsto \langle P'' \,;\, N\rangle$ *then:*

- $P' = P''$ *and* $N = C'[P]$ *for some* C'
- $\forall Q, \forall Q'\ \langle Q' \,;\, C[Q]\rangle \Longmapsto \langle Q' \,;\, C'[Q]\rangle$

The lemma follows by induction on the length of $\Longmapsto$. Suppose that $\langle P' \,;\, C[P]\rangle \longmapsto \langle P'' \,;\, N'\rangle \Longmapsto \langle P''' \,;\, N\rangle$. The result follows from the inductive hypothesis if we can prove that $P' = P''$ and $N' = C'[P]$ for some C' and $\forall Q, \forall Q'$, $\langle Q' \,;\, C[Q]\rangle \overset{\tau}{\longmapsto} \langle Q' \,;\, C'[Q]\rangle$. This in turn follows by induction on the derivation of $\langle P' \,;\, C[P]\rangle \longmapsto \langle P'' \,;\, N'\rangle$, since the derivation cannot consist of axioms Act or Write, and all the other rules (as well as axiom Read') satisfy the hypothesis.

Lemma 3.19. *If* $\langle P_1 \,;\, C[P_2]\rangle \overset{\sigma}{\Longmapsto} \langle P_1' \,;\, N\rangle$ *with* $\sigma \in \mathcal{A}^*$ *then:*

- *rule* Write *is not used*
- $N = C'[P_2]$ *with* $P_1 \xrightarrow{\sigma} P_1'$
- $\forall Q_1, Q_2$, *if* $Q_1 \xrightarrow{\sigma} Q_1'$ *then* $\langle Q_1 \,;\, C[Q_2]\rangle \overset{\sigma}{\Longmapsto} \langle Q_1' \,;\, C'[Q_2]\rangle$

The first item follows from the definition of the relation $\overset{\sigma}{\Longmapsto}$. The others are proved by induction on the length of the sequence σ. Let $\sigma = r\sigma'$ By Lemma 3.18, $\langle P_1 \,;\, C[P_2]\rangle \Longmapsto \langle P_1 \,;\, C'[P_2]\rangle \overset{r}{\longmapsto} \langle P_1' \,;\, N'\rangle \overset{\sigma'}{\Longmapsto} \langle P_1'' \,;\, N\rangle$, and $\forall Q_1, Q_2$, $\langle Q_1 \,;\, C[Q_2]\rangle \Longmapsto \langle Q_1 \,;\, C'[Q_2]\rangle$. If $\langle P_1 \,;\, C'[P_2]\rangle \overset{r}{\longmapsto} \langle P_1' \,;\, N'\rangle$, the derivation uses rule Act and by induction on the derivation we have that $C'[P_2]$ is of the form $C''[r?]$ for C'' a P-evaluation context, and $\langle P_1' \,;\, N'\rangle = \langle P_1' \,;\, C''[\texttt{true}]\rangle$ with $P_1 \xrightarrow{r} P_1'$, and $\forall Q_1, Q_2$, and for any Q_2-evaluation context C, if $Q_1 \xrightarrow{r} Q_1'$ then $\langle Q_1 \,;\, C[r?]\rangle \overset{r}{\longmapsto} \langle Q_1' \,;\, C[\texttt{true}]\rangle$. Then we can apply the inductive hypothesis to $\langle P_1' \,;\, N'\rangle \overset{\sigma'}{\Longmapsto} \langle P_1'' \,;\, N\rangle$ and the result follows.

Lemma 3.20. *If* $\langle P' \,;\, C[P]\rangle \overset{W}{\longmapsto} \langle P'' \,;\, N\rangle$ *then:*

- $C[P] = C'[loc := P]$ *and* C' *is a* P*-evaluation context*
- $P'' = P$ *and* $N = C'[\star]$
- $\forall Q, \forall Q'\ \langle Q' \,;\, C[Q]\rangle \overset{W}{\longmapsto} \langle Q \,;\, C''[\star]\rangle$ *and* $C'' = C'\{Q/P\}$

As above, the proof is by induction on the derivation of the reduction.

Finally, we note that if $\langle P_I \,;\, C[P]\rangle \Longrightarrow \langle P' \,;\, \texttt{true}\rangle$ then rule RefAct is never used in the derivation of the sequence of transitions, since whenever RefAct is used we either have `false` or a term that is stuck (and is not a value).

Suppose that $P \sim_{\mathrm{Tr}} Q$ and $\langle P_I \,;\, C[P]\rangle \Longrightarrow \langle P' \,;\, \texttt{true}\rangle$. It follows from the definition of $\longmapsto$ that we have two cases. If a W-labeled transition is never performed then by Lemma 3.19 we have $\langle P_I \,;\, C[Q]\rangle \Longrightarrow \langle P' \,;\, \texttt{true}\rangle$. If it is, then by Lemma 3.19 and Lemma 3.20

we derive $\langle P_I\,;\,C[P]\rangle \overset{\sigma_0}{\Longmapsto} \langle P'_I\,;\,C'[loc := P]\rangle \overset{W}{\longmapsto} \langle P\,;\,C'[\star]\rangle \overset{\sigma}{\Longmapsto} \langle P'\,;\,\texttt{true}\rangle$ with C' a P-evaluation context and σ of the form $W\sigma_1 W\sigma_2 W ... W\sigma_n$ for $\sigma_i \in \mathcal{A}^*$, and for all $i \geq 1$, $P \xrightarrow{\sigma_i}$. Hence, by the same lemmas and by the assumption that $P \sim_{\mathrm{Tr}} Q$ we derive that $\langle P_I\,;\,C[Q]\rangle \overset{\sigma_0}{\Longmapsto} \langle P'_I\,;\,C''[loc := Q]\rangle \overset{W}{\longmapsto} \langle Q\,;\,C''[\star]\rangle \overset{\sigma}{\Longmapsto} \langle Q'\,;\,\texttt{true}\rangle$, for $C'' = C'\{^{Q}\!/\!_{P}\}$, which in turn implies $\langle P_I\,;\,C[Q]\rangle \Longrightarrow \langle Q'\,;\,\texttt{true}\rangle$.

The proof of item (2) is analogous. We first assume that the reduction relation $\longrightarrow$ is based on the modified rule Read’, and then we define the reduction $\longmapsto$ as above, but adding the following clause: $\langle P\,;\,M\rangle \overset{\neg r}{\longmapsto} \langle P'\,;\,M'\rangle$ if the derivation of the transition uses the RefAct rule on label r. The set of negated action labels $\neg r$ is denoted by $\neg\mathcal{A}$.

Now, rule RefAct can be used in a sequence of reductions that reaches success. Hence, we have to modify Lemma 3.19 as follows: instead of traces σ_i, we use syntactic versions of failure traces, i.e., $\sigma_i \in (\mathcal{A} \cup \neg\mathcal{A})^*$, where $P \xrightarrow{\neg r} P'$ if $P' = P$ and $P \not\xrightarrow{r}$. Then the result follows as in the previous case, by considering processes P and Q that are failure trace equivalent, rather than trace equivalent.

Proof of Theorem 3.9 - right-to-left inclusions

We first prove the result for item (1), i.e., $\sim_{\mathrm{PB}} \subseteq \simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}}$.

In what follows, we use D, D' and their indexed versions to denote distributions on configurations of the form $\langle P\,;\,M\rangle$. For a distribution $\Delta = \sum_i p_i{\cdot}\texttt{dirac}(P_i)$ over processes, we also sometimes use $\langle\Delta\,;\,M\rangle$ to denote $\langle\Delta\,;\,\texttt{dirac}(M)\rangle$, i.e., the distribution $\sum_i p_i \cdot \texttt{dirac}(\langle P_i\,;\,M\rangle)$.

Analogously to the nondeterministic case, we first show that strong probabilistic bisimilarity is preserved by $\Lambda^{loc}_{\mathrm{V}}$-contexts, i.e., we prove that if $P \sim_{\mathrm{PB}} Q$ then $C[P] \sim_{\mathrm{PB}} C[Q]$, with $\sim_{\mathrm{PB}}$ defined on the transitions $\xrightarrow{\tau}$ (corresponding to $\longrightarrow$) and $\xrightarrow{\omega}$ (performed by the term $\texttt{true}$).

We show that the following is a probabilistic bisimulation:

$$\mathcal{R} \overset{\mathsf{def}}{=} \{(\langle P\,;\,C[\widetilde{P}]\rangle, \langle Q\,;\,C[\widetilde{Q}]\rangle) \ \mid\ P, \widetilde{P} \sim_{\mathrm{PB}} Q, \widetilde{Q}\}$$

where contexts C are polyadic.

To do so, we prove by induction on C that if $\langle P\,;\,C[\widetilde{P}]\rangle \mathcal{R} \langle Q\,;\,C[\widetilde{Q}]\rangle$ and $\langle P\,;\,C[\widetilde{P}]\rangle \longrightarrow D$ then $\langle Q\,;\,C[\widetilde{Q}]\rangle \longrightarrow D'$ with $D\,\texttt{lift}(\mathcal{R})\,D'$, which in turn means that there are an index set I and probability values $\{p_i\}_{i\in I}$ such that:

- $\Delta = \sum_i p_i \cdot \texttt{dirac}(\langle P_i\,;\,C_i[\widetilde{P_i}]\rangle)$;
- $\Theta = \sum_i p_i \cdot \texttt{dirac}(\langle Q_i\,;\,C_i[\widetilde{Q_i}]\rangle)$;
- $P_i \sim_{\mathrm{PB}} Q_i$ and $\widetilde{P_i} \sim_{\mathrm{PB}} \widetilde{Q_i}$

Most cases are proved analogously to the nondeterministic case, by either exploiting the fact that the transition reaches a Dirac distribution or by directly applying the inductive hypothesis. The interesting case is when the locations contain probabilistic bisimilar processes P and Q, and the terms perform an action test $r?$, i.e., $C[\widetilde{P}] = C[\widetilde{Q}] = r?$. There are two cases. If both P and Q do not perform r, then they reduce to distributions $\texttt{dirac}(\langle P\,;\,\texttt{false}\rangle)$ and $\texttt{dirac}(\langle Q\,;\,\texttt{false}\rangle)$, which are in $\texttt{lift}(\mathcal{R})$ since $\langle P\,;\,\texttt{false}\rangle \mathcal{R} \langle Q\,;\,\texttt{false}\rangle$. Otherwise, if $P \xrightarrow{r} \Delta$ and $Q \xrightarrow{r} \Theta$ then $\Delta\,\texttt{lift}(\sim_{\mathrm{PB}})\,\Theta$. Hence, there are an index set I and probability values $\{p_i\}_{i\in I}$ such that:

- $\Delta = \sum_i p_i \cdot \texttt{dirac}(P_i)$;
- $\Theta = \sum_i p_i \cdot \texttt{dirac}(Q_i)$;
- $P_i \sim_{\mathrm{PB}} Q_i$.

Since $\langle P\,;\, r?\rangle \longrightarrow \langle \Delta\,;\, \texttt{true}\rangle$ and $\langle Q\,;\, r?\rangle \longrightarrow \langle \Theta\,;\, \texttt{true}\rangle$, by applying the decomposition of Δ and Θ given above we derive that $\langle \Delta\,;\, \texttt{true}\rangle\, \texttt{lift}(\mathcal{R})\, \langle \Theta\,;\, \texttt{true}\rangle$.

Finally, probabilistic bisimilarity implies probabilistic trace equivalence, so we have $\langle P\,;\, C[\widetilde{P}]\rangle \sim_{\mathrm{PTr}} \langle Q\,;\, C[\widetilde{Q}]\rangle$, with traces defined on the labels τ and ω.

Since for any $P, \widetilde{P}, C$ the semantics of $\langle P\,;\, C[\widetilde{P}]\rangle$ is an RPLTS, and no state cannot perform both an ω and a τ action, we derive that the probability of success of $C[P]$ coincides with

$$\sum_{n\geq 0} \mathit{prob}(C[P], \tau^n \omega)$$

i.e., the sum of the probabilities of performing a τ trace of arbitrary length leading to success. Hence, it follows from $\langle P\,;\, C[\widetilde{P}]\rangle \sim_{\mathrm{PTr}} \langle Q\,;\, C[\widetilde{Q}]\rangle$ that they have the same probability of success.

The proof for $\Lambda^{loc}_{\mathrm{V-ref}}$ is the same, except for the case in which $C[\widetilde{P}] = C[\widetilde{Q}] = r?$ and both terms get stuck, since both the bisimilar processes P and Q in the locations cannot perform action r.

To prove the results for the inductive equivalences (items (2) and (3)), we redefine trace (and failure trace) equivalence as a relation on *subdistributions* over configurations of processes and terms, i.e., distributions of the form $\sum_i p_i \cdot \texttt{dirac}(\langle P_i\,;\, M_i\rangle)$ such that $\sum_i p_i$ is less than or equal to 1 (the difference being that the weights do not have to sum to 1).

Given a labeled transition relation $\xrightarrow{\mu}: S \to \mathcal{D}(S)$, we can lift it to subdistributions as follows:

$$\Delta \xrightarrow{\mu} \Delta' \text{ if } \Delta' = \sum_{P\in \mathrm{supp}(\Delta)^{\mu}} \Delta(P)\cdot \mathrm{der}(P)(\mu)$$

where $\mathrm{supp}(\Delta)^{\mu} = \{P \mid P \in \mathrm{supp}(\Delta) \wedge P \xrightarrow{\mu}\}$ and $\mathrm{der}(P)(\mu) = \Delta$ if $P \xrightarrow{\mu} \Delta$. For the sake of simplicity, we use the same symbol $\xrightarrow{\mu}$ for the lifted relation.

We extend the definition of the probability of σ-compatible computations to distributions as follows:

$$\mathit{prob}(\Delta,\sigma) = \begin{cases} 1 & \text{if } \mid \sigma \mid = 0 \\ \sum_{P\in\mathrm{supp}(\Delta)^{\mu}} \Delta(P)\cdot \mathit{prob}(\mathrm{der}(P)(\mu), \sigma') & \text{if } \sigma = \mu\sigma' \end{cases}$$

It is easy to see that $\mathit{prob}(\texttt{dirac}(P), \sigma) = \mathit{prob}(P, \sigma)$.

Let $\texttt{weight}(\sum_i p_i \cdot \texttt{dirac}(P_i)) = \sum_i p_i$ be the weight of a subprobability distribution. We can define probabilistic trace equivalence on subdistributions as follows:

$$\Delta \sim_{\mathrm{PTr}} \Theta \text{ if and only if } \forall \sigma \in \mathcal{A}^*,\ \texttt{weight}(\mathrm{der}(\Delta)(\sigma)) = \texttt{weight}(\mathrm{der}(\Theta)(\sigma))$$

Since $\mathit{prob}(\Delta, \sigma) = \texttt{weight}(\mathrm{der}(\Delta)(\sigma))$ for every trace σ, this relation coincides with probabilistic trace equivalence on states if we consider the Dirac distributions over the same states, i.e., $P \sim_{\mathrm{PTr}} Q$ if and only if $\texttt{dirac}(P) \sim_{\mathrm{PTr}} \texttt{dirac}(Q)$.

By considering transitions between subdistributions, we obtain the following useful property of probabilistic trace equivalence:

Lemma 3.21. *If* $\Delta \sim_{\mathrm{PTr}} \Theta$ *then:*

- $\mathtt{weight}(\Delta) = \mathtt{weight}(\Theta)$
- *if* $\Delta \xrightarrow{\mu} \Delta'$ *then* $\Theta \xrightarrow{\mu} \Theta'$ *and* $\Delta' \sim_{\mathrm{PTr}} \Theta'$

As in the proof for nondeterministic processes, we first consider a transition relation based on rule Read', since the probability of success of $C[P]$ is invariant with respect to this modification.

Then, analogously, we define a labeled transition $\xmapsto{\mu}$, for $\mu = W, r, \tau$, from configurations $\langle P\,;\, M\rangle$ to distributions of the form $D = \sum_i p_i \cdot \mathtt{dirac}(\langle P_i\,;\, M_i\rangle)$. We sometimes omit the Dirac function and write $\sum_i p_i \cdot \langle P_i\,;\, M_i\rangle$ to denote $\sum_i p_i \cdot \mathtt{dirac}(\langle P_i\,;\, M_i\rangle)$. This transition relation is lifted to a transition relation between subdistributions $D \xmapsto{\mu} D'$ as described above. Then, we define the weak version of this transition and we have the following lemmas.

Lemma 3.22. $\sum_i p_i \cdot \langle P_i\,;\, C[P]\rangle \xmapsto{\tau} D$ *implies:*

- $D = \sum_i p_i \cdot \langle P_i\,;\, C'[P]\rangle$
- $\forall Q, p'_j, Q_j,\ \sum_j p'_j \cdot \langle Q_j\,;\, C[Q]\rangle \xmapsto{\tau} \sum_j p'_j \cdot \langle Q_j\,;\, C'[Q]\rangle$

Lemma 3.23. $\sum_i p_i \cdot \langle P_i\,;\, C[P]\rangle \xmapsto{W} D$ *implies:*

- $C = C'[loc := [\cdot]]$ *for* C' *a* P*-evaluation context and* $D = \sum_i p_i \cdot \langle P\,;\, C'[\star]\rangle$
- $\forall Q, p'_j, Q_j,\ \sum_j p'_j \cdot \langle Q_j\,;\, C[Q]\rangle \xmapsto{W} \sum_j p'_j \cdot \langle Q\,;\, C''[\star]\rangle$*, for* $C'' = C'\{Q/P\}$

Lemmas 3.22 and 3.23 follow as in the nondeterministic case, by induction on the derivation of the transition.

Lemma 3.24. $\langle \Delta\,;\, C[P]\rangle \xmapsto{r} D$ *implies:*

- $D = \langle \mathrm{der}(\Delta)(r)\,;\, C'[P]\rangle$
- $\forall Q, \Theta$*, if* $\Delta \sim_{\mathrm{PTr}} \Theta$ *then* $\Theta \xrightarrow{r} \Theta'$ *for* $\Delta' \sim_{\mathrm{PTr}} \Theta'$ *and* $\langle \Theta\,;\, C[Q]\rangle \xmapsto{r} \langle \Theta'\,;\, C'[Q]\rangle$

By Lemma 3.21, we derive Lemma 3.24 as in Lemma 3.19 for the nondeterministic case, since $\langle \Delta\,;\, C[P]\rangle \xmapsto{r} D$ implies $C[P] = C'[r?]$ for C' a P-evaluation context. Hence, Lemma 3.25 follows by the definition of weak transition and by Lemmas 3.22 and 3.24.

Lemma 3.25. *For* $\sigma \in \mathcal{A}^*$*,* $\langle \Delta\,;\, C[P]\rangle \overset{\sigma}{\Longmapsto} D$ *implies:*

- $D = \langle \mathrm{der}(\Delta)(\sigma)\,;\, C[P]\rangle$
- $\forall Q, \Theta$*, if* $\Delta \sim_{\mathrm{PTr}} \Theta$ *then* $\Theta \xrightarrow{\sigma} \Theta'$ *for* $\Delta' \sim_{\mathrm{PTr}} \Theta'$ *and* $\langle \Theta\,;\, C[Q]\rangle \overset{\sigma}{\Longmapsto} \langle \Theta'\,;\, C'[Q]\rangle$

Using the reduction relations between subdistributions, by the Lemmas above we can see that: $\Longrightarrow$ and $\Longmapsto$ are deterministic, and $\langle \Delta\,;\,C[P]\rangle \longrightarrow D$ iff $\langle \Delta\,;\,C[P]\rangle \xmapsto{\mu} D$ with $\mu = W, r, \tau$ and D of the form $\langle \Delta'\,;\,C'[P]\rangle$,

Moreover, whenever $\sum_i p_i \cdot \langle P_i\,;\,C[P]\rangle \xmapsto{\mu} D$ with $\mu = W, \tau$ the whole distribution progresses (i.e., no configuration $\langle P_i\,;\,C[P]\rangle$ gets stuck or becomes true or false, and all $\langle P_i\,;\,C[P]\rangle$ perform a reduction), and whenever $\sum_i p_i \cdot \langle P_i\,;\,C[P]\rangle \xmapsto{r} D$ the configurations $\langle P_i\,;\,C[P]\rangle$ that do not progress are stuck (and will never return true or progress).

Hence, there is a unique distribution D of the form $\sum_i p_i \cdot \langle P_i\,;\,\texttt{true}\rangle$ such that $\langle \texttt{dirac}(P_I)\,;\,C[P]\rangle \Longrightarrow D$. The probability of success of $C[P]$ (given by the sum of the probabilities of all the computations reaching a state of the form $\langle P'\,;\,\texttt{true}\rangle$ for some P') can be equivalently defined as the weight of D.

If $\texttt{dirac}(\langle P_I\,;\,C[P]\rangle) \Longrightarrow \langle \Delta\,;\,\texttt{true}\rangle$ then, by Lemma 3.25, either $\texttt{dirac}(\langle P_I\,;\,C[P]\rangle) \overset{\sigma}{\Longmapsto} \langle \Delta\,;\,\texttt{true}\rangle$ and $\texttt{dirac}(\langle P_I\,;\,C[Q]\rangle) \overset{\sigma}{\Longmapsto} \langle \Delta\,;\,\texttt{true}\rangle$, or there exists a sequence σ of the form $\sigma_0 W \sigma_1 W \sigma_2 W ... W \sigma_n$ for $\sigma_i \in \{\mathcal{A}\}^*$ such that (by Lemma 3.23) $\langle P_I\,;\,C[P]\rangle \overset{\sigma_0}{\Longmapsto} \langle \Delta\,;\,C'[l := P]\rangle \overset{\sigma}{\Longmapsto} \langle \Delta'\,;\,\texttt{true}\rangle$, with C' a P-evaluation context, and $\langle P_I\,;\,C[Q]\rangle \overset{\sigma_0}{\Longmapsto} \langle \Delta\,;\,C''[l := Q]\rangle$, with $C'' = C'\{Q/P\}$.

If $\langle \Delta\,;\,C[P]\rangle \overset{\sigma}{\Longmapsto} D$ with σ of the form $\sigma_1 W \sigma_2 W ... W \sigma_n$, for $\sigma_i \in \{\mathcal{A}\}^*$, then by Lemmas 3.23 and 3.25 we have that $D = \langle \Delta'\,;\,C'[P]\rangle$ and there exists a sequence $\Delta_1, ..., \Delta_n$ such that $\Delta \xrightarrow{\sigma_1} \Delta_1$, $\Delta' = \Delta_n$ and, for $1 \leq i \leq n$, if $\Delta_i = \sum_j p_j \cdot P_j$ then Δ_{i+1} is the subdistribution such that $\sum_j p_j \cdot P \xrightarrow{\sigma_i} \Delta_{i+1}$. By the same lemmas, if $\Delta \sim_{\mathrm{PTr}} \Theta$ and $P \sim_{\mathrm{PTr}} Q$ then $\langle \Theta\,;\,C[Q]\rangle \overset{\sigma}{\Longmapsto} \langle \Theta'\,;\,C'[Q]\rangle$ and there is a sequence $\Theta_1, ..., \Theta_n$ with $\Theta \xrightarrow{\sigma_1} \Theta_1$, $\Theta' = \Theta_n$ and, for $1 \leq i \leq n$, if $\Theta_i = \sum_j p_j \cdot Q_j$ then $\sum_j p_j \cdot Q \xrightarrow{\sigma_i} \Theta_{i+1}$. Hence, Δ' and Θ' have the same weight, and the result follows.

For probabilistic failure trace equivalence, we extend the labels of the reduction relation $\longmapsto$ with actions in $\neg\mathcal{A}$. The proof is made more complicated by the fact that if we consider the definition of $\longmapsto$ on subdistributions, we have that the relation is now (externally) nondeterministic, since from a subdistribution $\langle \Delta\,;\,C[r?]\rangle$, for C an evaluation context, there are two reductions, one with label r and the other with label $\neg r$. However, these reductions capture the whole state space of the support of $\langle \Delta\,;\,C[r?]\rangle$, since either a state P performs r, in which case $\langle P\,;\,C[r?]\rangle \xmapsto{r} \langle \mathrm{der}(P)(r)\,;\,C[r?]\rangle$, or it does not, in which case $\langle P\,;\,C[r?]\rangle \xmapsto{\neg r} \langle \texttt{dirac}(P)\,;\,C[r?]\rangle$.

For $\Delta = \sum_i p_i \cdot P_i$, define $\Delta \xrightarrow{\neg r} \Delta'$ if $\Delta' = \sum_{\{i \mid P_i \in \mathrm{supp}(\Delta) \wedge P_i \not\xrightarrow{r}\}} p_i \cdot P_i$. Then $\Delta \sim_{\mathrm{PFTr}} \Theta$ is defined as $\Delta \sim_{\mathrm{PTr}} \Theta$, but using traces $\sigma \in (\mathcal{A} \cup \neg\mathcal{A})^*$.

Lemma 3.26. *$\langle \Delta\,;\,C[P]\rangle \xmapsto{\neg r} D$ implies:*

- $D = \langle \mathrm{der}(P)(\neg r)\,;\,C'[P]\rangle$
- $\forall Q, \Theta$*, if* $\Delta \sim_{\mathrm{PFTr}} \Theta$ *then* $\Theta \xrightarrow{\neg r} \Theta'$ *for* $\Delta' \sim_{\mathrm{PFTr}} \Theta'$ *and* $\langle \Theta\,;\,C[Q]\rangle \xmapsto{\neg r} \langle \Theta'\,;\,C'[Q]\rangle$

The lemma follows since $C[P]$ must be of the form $C'[r?]$ for C' a P-evaluation context, and if $\Delta \sim_{\mathrm{PFTr}} \Theta$ then their derivatives after performing r or $\neg r$ are still in $\sim_{\mathrm{PFTr}}$. Lemmas 3.22 and 3.23 remain the same, Lemma 3.24 holds with $\sim_{\mathrm{PFTr}}$ instead of $\sim_{\mathrm{PTr}}$, and Lemma 3.25 holds with $\sim_{\mathrm{PFTr}}$ instead of $\sim_{\mathrm{PTr}}$ and $\sigma \in (\mathcal{A} \cup \neg\mathcal{A})^*$.

The probability of success of $\langle P_I\,;\,C[P]\rangle$ is now given by the sum of the weights of the distributions Δ such that $\texttt{dirac}(\langle P_I\,;\,C[P]\rangle) \overset{\sigma}{\Longmapsto} \langle \Delta\,;\,\texttt{true}\rangle$, for all $\sigma \in (\mathcal{A} \cup \neg\mathcal{A} \cup \{W\})^*$

We show this by proving that for all n, $\langle \Delta\,;\, C[P]\rangle \Longrightarrow_n\xrightarrow{\omega} D$ iff

$$D = \sum_{\{\sigma\in(\mathcal{A}\cup\neg\mathcal{A}\cup\{W\})^* \mid \langle\Delta\,;\,C[P]\rangle\overset{\sigma}{\Longmapsto}_n\xrightarrow{\omega}D_\sigma\}} D_\sigma$$

This is well-defined since every σ denotes a unique path. The labels in $\mathcal{A}\cup\neg\mathcal{A}$ are the only ones that can create a branching in the labeled transition system with subdistributions on configurations as states and with transitions $\longmapsto$, where either D can perform only transition $\overset{\tau}{\longmapsto}$, or only transition $\overset{W}{\longmapsto}$, or both $\overset{r}{\longmapsto}$ and $\overset{\neg r}{\longmapsto}$, but with every state in the support performing either $\overset{r}{\longmapsto}$ or $\overset{\neg r}{\longmapsto}$, and not both. Hence, the set of all D' such that there is a path of length n with $D \overset{\mu_1}{\longmapsto} D_1 \overset{\mu_2}{\longmapsto} D_2 \ldots . \overset{\mu_n}{\longmapsto}\xrightarrow{\omega} D'$, for $\mu\in(\mathcal{A}\cup\neg\mathcal{A}\cup\{W,\tau\})$, coincides with the set of D' such that there is a $\sigma\in(\mathcal{A}\cup\neg\mathcal{A}\cup\{W\})^*$ such that $D \overset{\sigma}{\Longmapsto}_n\xrightarrow{\omega} D'$.

Proof of Theorem 3.9 - left-to-right inclusions

In order to prove $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V-ref}}} \subseteq\sim_{\mathrm{PB}}$ and $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{V}}} \subseteq\sim_{\mathrm{PB}}$, we exploit the testing characterization of probabilistic bisimilarity described in Section 2.3.3, using the language of tests $\mathbf{T}$. As we have seen, on RPLTS it holds that $P \sim_{\mathrm{PB}} Q$ iff $\mathrm{Pr}(\mathbf{t},P) = \mathrm{Pr}(\mathbf{t},Q)$ for every test $\mathbf{t}$ in $\mathbf{T}$ [BMOW05]. We show that these tests are encodable in $\Lambda^{loc}_{\mathrm{V-ref}}$ i.e., that there is an encoding $\mathrm{Enc}() : \mathbf{T} \to \Lambda^{loc}_{\mathrm{V-ref}}$ such for every test $\mathbf{t}$ in $\mathbf{T}$ and for every P, $\mathrm{Pr}(\mathbf{t},P) = \mathit{prob}(\mathcal{SC}(\langle P\,;\,\mathrm{Enc}(\mathbf{t})\rangle))$. Hence, if $P \not\sim_{\mathrm{PB}} Q$ then there is context of $\Lambda^{loc}_{\mathrm{V-ref}}$ (namely, context $C \overset{\mathtt{def}}{=} (\mathit{loc} := [\cdot])\,\underline{\mathtt{seq}}\,\mathrm{Enc}(\mathbf{t})$, for some $\mathbf{t}$) that distinguishes P and Q. The same holds for the language $\Lambda^{loc}_{\mathrm{V}}$, since it includes $\Lambda^{loc}_{\mathrm{V-ref}}$, and since the the possibility of exploring the `else` branches does not allow to add any successful computation (since the `else` branches always lead to `false`).

We define the encoding analogously to the encoding of the Simulation Logic Formulas in the nondeterministic case:

$$\begin{aligned}
&\mathrm{Enc}(\omega) = \mathtt{true}\\
&\mathrm{Enc}(r.\mathbf{t}) = \mathtt{if}\ r?\ \mathtt{then}\ \mathrm{Enc}(\mathbf{t})\ \mathtt{else\ false}\\
&\mathrm{Enc}((\mathbf{t}_1,\mathbf{t}_2)) = (\lambda z.\,\mathtt{if}\ \ \mathrm{Enc}(\mathbf{t}_1)\ \ \mathtt{then}\ \ (\mathit{loc} := z)\,\underline{\mathtt{seq}}\,\mathrm{Enc}(\mathbf{t}_2)\ \ \mathtt{else\ false})!\mathit{loc}
\end{aligned}$$

We prove by induction on the definition of $\mathbf{t}$ that $\mathrm{Pr}(\mathbf{t},P) = \mathit{prob}(\mathcal{SC}(\langle P\,;\,\mathrm{Enc}(\mathbf{t})\rangle))$. The case $\mathbf{t} = \omega$ is trivial.

- Case $\mathbf{t} = r.\mathbf{t}'$. The interesting case is when $P \xrightarrow{a} \Delta$. Then the result follows from the fact that the set $\mathcal{SC}(\langle P\,;\,\mathrm{Enc}(\mathbf{t})\rangle)$ coincides with the set of computations of the form

$$\begin{aligned}
&\langle P\,;\,\mathrm{Enc}(\mathbf{t})\rangle,\\
&\langle P'\,;\,\mathtt{if\ true\ then}\ \mathrm{Enc}(\mathbf{t}')\ \mathtt{else\ false}\rangle,\\
&\langle P'\,;\,\mathrm{Enc}(\mathbf{t}')\rangle,\\
&c'
\end{aligned}$$

for $P' \in \mathrm{supp}(\Delta)$ and $c' \in \mathcal{SC}(\langle P' \,;\, \mathrm{Enc}(\mathbf{t}')\rangle)$. We have

$$\begin{aligned}
\mathit{prob}(\mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t})\rangle)) &= \textstyle\sum_{c \in \mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t})\rangle)} \mathit{prob}(c) \\
&= \textstyle\sum_{P' \in \mathrm{supp}(\Delta)} \Delta(P') \cdot \sum_{c \in \mathcal{SC}(\langle P' \,;\, \mathrm{Enc}(\mathbf{t}')\rangle)} \mathit{prob}(c) \\
&= \textstyle\sum_{P' \in \mathrm{supp}(\Delta)} \Delta(P') \cdot \mathrm{Pr}(\mathbf{t}', P') \\
&= \mathrm{Pr}(\mathbf{t}, P)
\end{aligned}$$

- Case $\mathbf{t} = (\mathbf{t}_1, \mathbf{t}_2)$. The set of computations $\mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t})\rangle)$ coincides with the set of computations of the form

$$\begin{aligned}
&\langle P \,;\, (\lambda z.\, \mathtt{if}\ \ \mathrm{Enc}(\mathbf{t}_1)\ \ \mathtt{then}\ \ (\mathit{loc} := z)\, \underline{\mathtt{seq}}\, \mathrm{Enc}(\mathbf{t}_2)\ \ \mathtt{else}\ \mathtt{false})!\mathit{loc}\rangle, \\
&\langle P \,;\, (\lambda z.\, \mathtt{if}\ \ \mathrm{Enc}(\mathbf{t}_1)\ \ \mathtt{then}\ \ (\mathit{loc} := z)\, \underline{\mathtt{seq}}\, \mathrm{Enc}(\mathbf{t}_2)\ \ \mathtt{else}\ \mathtt{false})P\rangle, \\
&C[c_1], \\
&\langle P' \,;\, \mathit{loc} := P\, \underline{\mathtt{seq}}\, \mathrm{Enc}(\mathbf{t}_2)\rangle, \\
&\langle P \,;\, \star\, \underline{\mathtt{seq}}\, \mathrm{Enc}(\mathbf{t}_2)\rangle, \\
&c_2
\end{aligned}$$

for $c_i \in \mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t}_i)\rangle)$, and $C[c_1]$ denoting the computation c_1, but with every term put in the context $C = \mathtt{if}\ [\cdot]\ \mathtt{then}\ (\mathit{loc} := P)\, \underline{\mathtt{seq}}\, \mathrm{Enc}(\mathbf{t}_2)\ \mathtt{else}\ \mathtt{false}$, and P' the last value of the location in c_1. Hence, the result follows from the inductive hypothesis on $\mathbf{t}_1$ and $\mathbf{t}_2$:

$$\begin{aligned}
\mathit{prob}(\mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t})\rangle)) &= \textstyle\sum_{c \in \mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t})\rangle)} \mathit{prob}(c) \\
&= \textstyle\sum_{c_1 \in \mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t}_1)\rangle)} \mathit{prob}(c_1) \cdot \sum_{c_2 \in \mathcal{SC}(\langle P \,;\, \mathrm{Enc}(\mathbf{t}_2)\rangle)} \mathit{prob}(c_2) \\
&= \mathrm{Pr}(\mathbf{t}_1, P) \cdot \mathrm{Pr}(\mathbf{t}_2, P) \\
&= \mathrm{Pr}(\mathbf{t}, P)
\end{aligned}$$

For the inductive equivalences, we first consider consider probabilistic failure trace equivalence, i.e., we prove $\simeq^{\mathcal{L}}_{\Lambda^{loc}_{\mathrm{N}}} \subseteq \sim_{\mathrm{PFTr}}$. For $\sigma \in (\mathcal{A} \cup \neg\mathcal{A})^*$, we define the term t_σ as follows by induction on σ:

- if σ is empty then $t_\sigma = \mathtt{true}$
- if $\sigma = r\sigma'$ then $t_\sigma = \mathtt{if}\ \ r?\ \ \mathtt{then}\ \ t_{\sigma'}\ \ \mathtt{else}\ \mathtt{false}$
- if $\sigma = \neg r\sigma'$ then $t_\sigma = \mathtt{if}\ \ r?\ \ \mathtt{then}\ \mathtt{false}\ \mathtt{else}\ \ t_{\sigma'}$

We exploit the definitions of reductions and labeled transition relations $\xmapsto{\mu}$ used in the opposite direction of the proof, and the definition of failure trace equivalence as a relation on subdistributions.

We prove by induction on σ that $\Delta \xrightarrow{\sigma} \Delta'$ iff $\langle \Delta \,;\, t_\sigma\rangle \overset{\sigma}{\Longmapsto} \langle \Delta' \,;\, \mathtt{true}\rangle$.

At each step from $\langle \Delta \,;\, t_\sigma\rangle$ either we have a reduction $\overset{\tau}{\Longmapsto}$ where the whole distribution progresses or we have a branch with $\xmapsto{r}$ and $\xmapsto{\neg r}$, where one of the two branches deterministically progresses to $\mathtt{false}$. Hence, there is only one $\longmapsto$-path from $\langle \Delta \,;\, t_\sigma\rangle$ reaching a distribution with states with values $\mathtt{true}$, and this distribution is exactly the one such that $\langle \Delta \,;\, t_\sigma\rangle \overset{\sigma}{\Longmapsto} \langle \Delta' \,;\, \mathtt{true}\rangle$. As a consequence, the probability of success of $\langle \Delta \,;\, t_\sigma\rangle$ is the weight of Δ', which in turn coincides with the probability of Δ of performing trace σ.

We derive that

$$\begin{aligned}
prob(P,\sigma) &= prob(\texttt{dirac}(P),\sigma)\\
&= \texttt{weight}(\mathrm{der}(\sigma)(\texttt{dirac}(P)))\\
&= \texttt{weight}(\Delta' \mid \langle \texttt{dirac}(P)\,;\,t_\sigma\rangle \overset{\sigma}{\Longmapsto} \langle \Delta'\,;\,\texttt{true}\rangle)\\
&= prob(\mathcal{SC}(\langle P\,;\,t_\sigma\rangle))\\
&= prob(\mathcal{SC}(\langle P_I\,;\,loc := P\,\underline{\texttt{seq}}\,t_\sigma\rangle))
\end{aligned}$$

Hence, processes with different probabilities of performing a failure trace σ (which is a special case of a sequence $\sigma \in (\mathcal{A} \cup \neg\mathcal{A})^*$) have different probabilities of success when put in context $loc := [\cdot]\,\underline{\texttt{seq}}\,t_\sigma$.

For trace equivalence the proof is analogous, but we only consider tests t_σ with $\sigma \in \mathcal{A}^*$. At each step from $\langle \Delta\,;\,t_\sigma\rangle$ either we have a reduction $\Longmapsto$ where the whole distribution progresses or we have a reduction $\overset{r}{\Longmapsto}$ where the part of distribution that does not progress gets stuck (and therefore never reaches a `true` state).

Proof of Theorem 3.11 - left-to-right inclusions

For the coinductive equivalences, as in the proof for λ-calculi, we show that the tests for Ready Simulation logic and Simulation logic can be encoded in $\mathrm{HO}\pi_{\texttt{pass},\mathrm{ref}}$ and $\mathrm{HO}\pi_{\texttt{pass}}$ respectively.

We define the encoding as follows:

$$\begin{aligned}
&\mathrm{Enc}(\top) = \omega\\
&\mathrm{Enc}(\neg r) = \widetilde{r}_l.\omega\\
&\mathrm{Enc}(\langle\, r\,\rangle F) = \overline{r}.\mathrm{Enc}(F)\\
&\mathrm{Enc}(F_1 \wedge F_2) = \texttt{pass}_l(x).(\llbracket x\rrbracket_l \mid \mathrm{Enc}(F_1)\{{\texttt{pass}_l(y).(\llbracket x\rrbracket_l \mid \mathrm{Enc}(F_2))}/{\omega}\}
\end{aligned}$$

For $C = \llbracket[\cdot]\rrbracket_l \mid \mathrm{Enc}(F)$, we have that $C[P] \Longrightarrow\xrightarrow{\omega}$ iff $P \models F$.

For failure trace equivalence, we prove that for any failure trace $\phi = (r_i, F_i)_{i\leq n}$, process P has the failure trace ϕ if and if $P \mid t_\phi \Longrightarrow\xrightarrow{\omega}$, where t_ϕ is defined by induction on n:

$$t_\emptyset = \omega \qquad t_{(r_i,F_i)_{i\leq n+1}} = r_{n+1}.t_{F_{n+1}}.t_{(r_i,F_i)_{i\leq n}}$$

and t_F is ω if F is empty, while for $r \in F$ and $F' = F \setminus \{r\}$ we have $t_F = \widetilde{r}_l.t_{F'}$.

All these tests are sequential, hence the proofs follow analogously to the λ-calculi.

Proof of Theorem 3.11 - right-to-left inclusions

The proof structure is analogous to the λ-calculus case.

We start from ready simulation equivalence and $\mathrm{HO}\pi_{\texttt{pass}\mathrm{ref}}$. In what follows, we write $\mu \in \mathcal{A}^F$ if $\mu = \tau \;\big|\; r \;\big|\; \omega \;\big|\; \overline{\widetilde{r}_l} \;\big|\; \widetilde{r}_l$ (as usual, r denotes action $\overline{r}\langle\star\rangle$) and $\mu \in \mathcal{A}^H$ if $\mu = \overline{a}\langle M\rangle \;\big|\; a\langle M\rangle \;\big|\; \overline{\texttt{pass}_l}M \;\big|\; \texttt{pass}_l M$. We first show that ready simulation equivalence is preserved by (pure) $\mathrm{HO}\pi_{\texttt{pass},\mathrm{ref}}$-contexts, by proving that the relation

$$\mathcal{R} \overset{\texttt{def}}{=} \{(C[\widetilde{P}], C[\widetilde{Q}]) \;\mid\; \widetilde{P} \text{ is ready simulated by } \widetilde{Q}\}$$

where C is a polyadic context of $\mathrm{HO}\pi_{\texttt{pass},\mathrm{ref}}$, is a strong ready simulation on transitions labeled by first-order actions $\mathcal{A}^F$. As a consequence, since ready simulation equivalence implies trace equivalence, any successful computation (i.e., any trace of the form $\tau^n\omega$) from $C[P]$ may be mimicked by $C[Q]$, and vice-versa.

The proof uses the following syntactical characterization of the shape of processes performing an higher-order action, i.e., an action in $\mathcal{A}^H$.

Lemma 3.27. *If $C[\widetilde{P}] \xrightarrow{\mu} M'$, for $\mu \in \mathcal{A}^H$, then $C = C_{l_1}[...C_{l_n}[C_{l_{n+1}}]...]$ for some n, where:*

- *$C_{l_i} = C^1_{l_i} \mid [\![[\cdot]]\!]_{l_i} \mid C^2_{l_i}$ for $i \leq n$, or a variant where one or both $C^1_{l_i}$ and $C^2_{l_i}$ (and the parallel composition) do not occur and such that $[\cdot]$ is a hole labeled so as to be filled with context $C_{l_{i+1}}$,*
- *$C_{l_{n+1}} = C^1_{l_{n+1}} \mid \mu'.C' \mid C^2_{l_{n+1}}\mu'.C'$, where $\mu' = \mu$ if $\mu' = \overline{a}\langle C'[\widetilde{P'}]\rangle$ or $\mu' = \overline{\mathtt{pass}_{l_{i+1}}}C'[\widetilde{P'}]$, and $\mu' = \mathtt{pass}_l(x)$ if $\mu = \mathtt{pass}_l N$ for some N, and $\mu' = a(x)$ if $\mu = a\langle N\rangle$ for some N.*

The lemma follows by induction on the derivation of $C[\widetilde{P}] \xrightarrow{\mu} M'$.

Then, we prove the result by structural induction on the contexts C of $\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}$. Let $C[\widetilde{P}]\,\mathcal{R}\,C[\widetilde{Q}]$. We show that:

1. for all $\mu \in \mathcal{A}^F$, if $C[\widetilde{P}] \xrightarrow{\mu} M$ then $C[\widetilde{Q}] \xrightarrow{\mu} M'$ and $M\,\mathcal{R}\,M'$, for some M',
2. for all $\mu \in \mathcal{A}^F$, if $C[\widetilde{Q}] \xrightarrow{\mu}$ then $C[\widetilde{P}] \xrightarrow{\mu}$.

- if $C = [\cdot]$, the result follows by $C[P] = P \lesssim_{\mathrm{RS}} Q = C[Q]$.
- if $C = \omega.C'$, $C = a(x).C'$, $C = \widetilde{r}_l.C'$, $C = \overline{a}\langle N\rangle.C'$, $C = \mathtt{pass}_l(x).C'$ and the action performed is first-order, then the reached processes are $C'[\widetilde{P}]$ and $C'[\widetilde{Q}]$, and for all $\mu \in \mathcal{A}^F$, $C[\widetilde{P}] \xrightarrow{\mu}$ iff $C[\widetilde{Q}] \xrightarrow{\mu}$.
- if $C = [\![C']\!]_l$, there are two cases:
  - $C'[\widetilde{P}] \xrightarrow{\mu} M'$ with $\mu \in \mathcal{A}^F$, and the transition is derived by rule Kell. Then the result follows by the inductive hypothesis.
  - $C[\widetilde{P}] \xrightarrow{\mu} M$ with $\mu = \widetilde{r}_l$ is derived by rule RefLoc. Then $C'[\widetilde{P}] = P \not\xrightarrow{\mu}$ and $C'[\widetilde{Q}] = Q \not\xrightarrow{\mu}$. Hence, $M = [\![P]\!]_l$, $M' = [\![Q]\!]_l$ and the result follows.

  The second condition follows analogously, by considering the two cases above when deriving $C[\widetilde{Q}] \xrightarrow{\mu}$ and using the fact that $P \lesssim_{\mathrm{RS}} Q$ and $Q \xrightarrow{\mu}$ imply $P \xrightarrow{\mu}$
- if $C = C_1 \mid C_2$, then $C[\widetilde{P}] = C_1[\widetilde{P}] \mid C_2[\widetilde{P}]$ and $C[\widetilde{Q}] = C_1[\widetilde{Q}] \mid C_2[\widetilde{Q}]$ with $\widetilde{P}_i \lesssim_{\mathrm{RS}} \widetilde{Q}_i$. There are three main cases:
  - $C[\widetilde{P}] \xrightarrow{\mu} M$ is derived by rule ParL (the case of ParR is symmetric). Then the result follows by the inductive hypothesis.
  - $C[\widetilde{P}] \xrightarrow{\tau} M$ is derived by rule Comm, by the synchronization of actions $\mu, \overline{\mu}$ in $\mathcal{A}^F$. The result follows again by the inductive hypothesis.

- $C[\widetilde{P}] \xrightarrow{\tau} M$ is derived by rule Comm, by the synchronization of actions $\mu, \overline{\mu}$ in $\mathcal{A}^H$. Then we cannot use the inductive hypothesis, and the result follows from Lemma 3.27.

Suppose now that $C[\widetilde{Q}] \xrightarrow{\mu}$. Then we have the same three cases, and the result follows analogously.

The proof for HOπ_{pass} is analogous, but we consider contexts filled with processes such that $\widetilde{P} \lesssim_{\mathrm{S}} \widetilde{Q}$, and we only prove that the obtained relation is a simulation on actions in $\mathcal{A}^F$.

Then the result follows since actions τ and ω are included in $\mathcal{A}^F$, and ready similarity implies similarity, which implies trace inclusion. Hence, if $P \lesssim_{\mathrm{RS}} Q$, or $P \lesssim_{\mathrm{S}} Q$, and $C[P]$ performs trace $\tau^n\omega$ for some n (i.e., it has a successful computation) then $C[Q]$ performs trace $\tau^n\omega$.

For the inductive equivalences, as in the case of λ-calculi, we consider an alternative labeled semantics which is meant to keep track of when derivatives of the initial processes are tested for actions.

Define the following:

- a process P in $\mathcal{L}$ is active in a term M of HO$\pi(\mathcal{L})$ if $M = M_1 \mid M_2 \mid ... \mid M_2$ and $P = M_i$ for some i.
- we say that C is a P-context of HOπ if it is a context where the only process name in $\mathcal{L}$ that occurs is P, and P is inactive (i.e., the context is not a parallel composition where P occurs top level). We denote such a context with C_P, and we use C_Q for the same context with P substituted to Q.

Let C_P be a P-context and let $\widetilde{S}$ be a sequence of processes in $\mathcal{L}$. Let $(\widetilde{S})_i$ denote the i-th element of the sequence. We assume that every hole of a context is numbered and occurs at most once. We write $\widetilde{S} \xrightarrow{r_i} \widetilde{S'}$ if $(\widetilde{S})_i \xrightarrow{r} P'$ and $\widetilde{S'}$ is equal to $\widetilde{S}$ but with P' at the i-th place. We extend this notation to the case where more than one action is performed in parallel, i.e., $\widetilde{S} \xrightarrow{\mathbf{r}} \widetilde{S'}$ for $\mathbf{r}$ a set of indexed labels $r_i, r'_j, ...$, each with different indexes (and where the labels are indexed in $\widetilde{S}$, i.e., they have as maximal index the length $\mid \widetilde{S} \mid$ of the sequence).

If $C_P[\widetilde{S}] \xrightarrow{\tau} M'$ then it follows from the operational semantics of HOπ that $M' = C'_P[\widetilde{S'}, P^n]$, where $\widetilde{S} \xrightarrow{\mathbf{r}} \widetilde{S'}$ for some (possibly empty) set $\mathbf{r}$ of labels indexed in $\widetilde{S}$, and P^n is a sequence composed by n-copies of P.

This holds since a synchronization might bring to the top level some copies of P that occur in C_P (that is, the sequence P^n), and since it can be be given by three possible kinds of interactions between processes at the top level:

- two prefixes of HOπ synchronize, and then $\mathbf{r} = \emptyset$
- there is a synchronization where process S_i of $\widetilde{S}$ performing r synchronizes with a prefix $\bar{r}$ in the context, in which case $\mathbf{r} = \{r_i\}$
- there are processes $S_i \xrightarrow{r}$ and $S_j \xrightarrow{\bar{r}}$ in $\widetilde{S}$ that synchronize with each other, in which case $\mathbf{r} = \{r_i, \bar{r}_j\}$.

Moreover, for any Q and $\widetilde{T}$ such that $\widetilde{T} \xrightarrow{\mathbf{r}} \widetilde{T'}$ we have $C_Q[\widetilde{T}] \xrightarrow{\tau} C'_Q[\widetilde{T'}, Q^n]$.

Since the three cases described above cover all cases in which $C_P[\widetilde{S}]$ performs a τ action, we can derive by induction on the length n of the τ-labeled sequence $C_P[\widetilde{S}] \Longrightarrow M'$ that $C_P[\widetilde{S}] \Longrightarrow M'$ iff there are $C_{i\,P}[\widetilde{S_i}]$ for $0 \leq i \leq n$ such that:

- $\widetilde{S}_i \xrightarrow{\mathbf{r}_{i+1}} \widetilde{S}'_{i+1}$
- $\widetilde{S}_{i+1} = \widetilde{S}'_{i+1}, \widetilde{P}$
- $C_{0\,P}[\widetilde{S_0}] = C_P[\widetilde{S}]$
- $M' = C_{n\,P}[\widetilde{S_n}]$

We write $C_P[\widetilde{S}] \xmapsto{\mathbf{r}} C'_P[\widetilde{S'}, P^n]$ if $C_P[\widetilde{S}] \xrightarrow{\tau} C'_P[\widetilde{S'}, P^n]$ with $\widetilde{S} \xrightarrow{\mathbf{r}} \widetilde{S'}$, and we write $C_P[\widetilde{S}] \overset{\widetilde{\mathbf{r}}}{\Longmapsto} M'$ for its reflexive and transitive closure. Then, $C_P[\widetilde{S}] \Longrightarrow M'$ iff $C_P[\widetilde{S}] \overset{\widetilde{r}}{\Longmapsto} M'$, with $\widetilde{S} \xrightarrow{\widetilde{r}}$.

Note that the actions in $\widetilde{r}$ might be indexed with indexes that are greater than $\mid \widetilde{S} \mid$. Then $\widetilde{S} \xrightarrow{\widetilde{r}}$ is defined by restricting the set indexes to those of $\widetilde{S}$. More formally, given a sequence $\widetilde{\mathbf{r}}$ of sets of actions with indexes in n, each of which contains at most one action for each index, let $\#_i(\widetilde{\mathbf{r}})$ be the sequence of actions with index i. Then $\widetilde{S} \xrightarrow{\widetilde{r}}$ iff $\forall i \leq\mid \widetilde{S} \mid$, $(\widetilde{S})_i \xrightarrow{\#_i(\widetilde{\mathbf{r}})}$.

Lemma 3.28. *If $C_P[\widetilde{S}] \overset{\widetilde{\mathbf{r}}}{\Longmapsto}\xrightarrow{\omega}$ then for any Q such that $P \sim_{\mathrm{Tr}} Q$ and for any $\widetilde{T}$ of the same length as $\widetilde{S}$ such that $(\widetilde{T})_i \xrightarrow{\#_i\widetilde{\mathbf{r}}}$ for every index $i \leq\mid \widetilde{S} \mid$ we have that $C_Q[\widetilde{T}] \overset{\mathbf{r}}{\Longmapsto} C'_Q[\widetilde{T'}]$.*

We prove the lemma by induction on the length of $\widetilde{\mathbf{r}}$. For the inductive case, suppose $C_P[\widetilde{S}] \xmapsto{\mathbf{r}} M' \overset{\widetilde{\mathbf{r}}'}{\Longmapsto}\xrightarrow{\omega}$. Then (by induction on the derivation of the transition) $M' = C'_P[\widetilde{S'}]$ with $\widetilde{S} \xrightarrow{\mathbf{r}'} \widetilde{S''}$ and $\widetilde{S'} = \widetilde{S''}, \widetilde{P}$, and $C_Q[\widetilde{T}] \xmapsto{\mathbf{r}} C'_Q[\widetilde{T''}, \widetilde{Q}]$, with $\widetilde{T} \xrightarrow{\mathbf{r}} \widetilde{T''}$ and $(\widetilde{T''})_i \xrightarrow{\#_i\widetilde{\mathbf{r}}}$ for $i \leq\mid \widetilde{S} \mid=\mid \widetilde{S'_1} \mid$. Then, since P and Q are trace equivalent, $\widetilde{S''}, \widetilde{P} \xrightarrow{\widetilde{\mathbf{r}}'}$ implies $\widetilde{T''}, \widetilde{Q} \xrightarrow{\widetilde{\mathbf{r}}'}$. Therefore, we can apply the inductive hypothesis to $C'_Q[\widetilde{T''}, \widetilde{Q}]$ and derive that $C_Q[\widetilde{T}] \xmapsto{\mathbf{r}} C'_Q[\widetilde{T''}, \widetilde{Q}] \overset{\widetilde{\mathbf{r}}'}{\Longmapsto}\xrightarrow{\omega}$.

For failure traces, the proof is the same but the sets $\mathbf{r}$ now include not only indexed actions r_i but also indexed refusal actions $\neg r_i$, which correspond to the case when the top-level process P_i cannot perform action i and synchronizes with a refusal operator on r.

Proof of Theorem 3.13

To prove item (1), we first show that the tests characterizing probabilistic bisimilarity of Section 2.3.3 can be encoded in HOπ, as in the proof for call-by-value λ-calculi.

We define the encoding as follows:

$$\mathrm{Enc}(\omega) = \omega$$
$$\mathrm{Enc}(r.\mathbf{t}) = \overline{r}.\mathrm{Enc}(\mathbf{t})$$
$$\mathrm{Enc}((\mathbf{t}_1, \mathbf{t}_2)) = \mathtt{pass}_l(x).(\llbracket x \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}_1)\{{}^{\mathtt{pass}_l(y).(\llbracket x \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}_2))}/\omega\}$$

For any test $\mathbf{t}$, since the encoding of $\mathbf{t}$ is sequential we have that the process $\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t})$ has no internal nondeterminism (with respect to τ-actions). Hence, there is only one possible resolution of $\mathcal{SC}(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}))$, for any P, and the probability of success is unique (i.e., may and must success coincide).

We prove by induction on the definition of $\mathbf{t}$ that $\mathrm{Pr}(\mathbf{t}, P) = \mathit{prob}(\mathcal{SC}(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}))$. The case $\mathbf{t} = \omega$ is trivial.

- Case $\mathbf{t} = r.\mathbf{t}'$. The interesting case is when $P \xrightarrow{r} \Delta$. Then the result follows from the fact that the set $\mathcal{SC}(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}))$ coincides with the set of computations of the form
$$\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}),\ \llbracket P' \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}'),\ c'$$
for $P' \in \mathrm{supp}(\Delta)$ and $c' \in \mathcal{SC}(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}))$. We have
$$\begin{aligned} \mathit{prob}(\mathcal{SC}(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}))) &= \textstyle\sum_{c \in \mathcal{SC}(\llbracket P \rrbracket_l \,\mid\, \mathrm{Enc}(\mathbf{t}))} \mathit{prob}(c) \\ &= \textstyle\sum_{P' \in \mathrm{supp}(\Delta)} \Delta(P') \cdot \sum_{c \in \mathcal{SC}(\llbracket P' \rrbracket_l \,\mid\, \mathrm{Enc}(\mathbf{t}'))} \mathit{prob}(c) \\ &= \textstyle\sum_{P' \in \mathrm{supp}(\Delta)} \Delta(P') \cdot \mathrm{Pr}(\mathbf{t}', P') \\ &= \mathrm{Pr}(\mathbf{t}, P) \end{aligned}$$

- Case $\mathbf{t} = (\mathbf{t}_1, \mathbf{t}_2)$. The result follows form the fact that the set of computations $\mathcal{SC}(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}))$ coincides with the set of computations of the form
$$\begin{aligned} &\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}), \\ &\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}_1)\{^{\mathtt{pass}_l(y).(\llbracket P \rrbracket_l \,\mid\, \mathrm{Enc}(\mathbf{t}_2))}/_{\omega}\}, \\ &c_1, \\ &\llbracket P' \rrbracket_l \mid \mathtt{pass}_l(y).(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}_2)), \\ &c_2, \\ &\omega \end{aligned}$$
for $c_i \in \mathcal{SC}(\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t}_i))$ but without the last term, and P' the last value of the location in c_1 (it is easy to prove by induction on $\mathbf{t}$ that c is a successful computation from $\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t})$ iff it is a τ-labeled computation from $\llbracket P \rrbracket_l \mid \mathrm{Enc}(\mathbf{t})$ whose last term is of the form $\llbracket P' \rrbracket_l \mid \omega$).

Hence, we have $\simeq^{\mathcal{L}}_{\mathrm{HO}\pi_{\mathtt{pass}}} \subseteq \sim_{\mathrm{PB}}$. Since $\mathrm{HO}\pi_{\mathtt{pass}}$ is a sublanguage of $\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}$, we also have $\simeq^{\mathcal{L}}_{\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}} \subseteq \simeq^{\mathcal{L}}_{\mathrm{HO}\pi_{\mathtt{pass}}}$.

It remains to prove that $\sim_{\mathrm{PB}} \subseteq \simeq^{\mathcal{L}}_{\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}}$.

As in the nondeterministic case, and like for λ-calculi, we first show that probabilistic bisimilarity is preserved by contexts of $\mathrm{HO}\pi_{\mathtt{pass},\mathrm{ref}}$, with respect to labels τ and ω.

Formally, we show that the following is a probabilistic bisimulation on the first-order labels (defined as in the proof of the corresponding item in Theorem 3.11):
$$\mathcal{R} \overset{\mathtt{def}}{=} \{(C[\widetilde{P}], C[\widetilde{Q}]) \mid \widetilde{P} \sim_{\mathrm{PB}} \widetilde{Q}\}$$
where contexts are polyadic. We prove by induction on C that if $C[\widetilde{P}]\ \mathcal{R}\ C[\widetilde{Q}]$ and $C[\widetilde{P}] \xrightarrow{\mu} \Delta$ for $\mu \in \mathcal{A}^F$ then $C[\widetilde{Q}] \xrightarrow{\mu} \Theta$ with $\Delta\, \mathtt{lift}(\mathcal{R})\, \Theta$.

The proof proceeds as in the corresponding case in the proof for the nondeterministic case (Theorem 3.11). Whenever a higher-order action is performed, we have that the analogous of Lemma 3.27 holds, and performing the higher-order action leads to a dirac distribution.

We only consider the case when $C = C_1 \mid C_2$ and $C[\widetilde{P}] \xrightarrow{\tau} \Delta = \Delta_1 \mid \Delta_2$, derived by rule Comm with a synchronization of $C_1[\widetilde{P}] \xrightarrow{\mu} \Delta_1$ and $C_2[\widetilde{P}] \xrightarrow{\mu} \Delta_2$, for $\mu \in \mathcal{A}^F$. Then $C[\widetilde{Q}] = C_1[\widetilde{Q}] \mid C_2[\widetilde{Q}]$ with $\widetilde{P} \sim_{\mathrm{PB}} \widetilde{Q}$, and by the inductive hypothesis $C_1[\widetilde{Q}] \xrightarrow{\mu} \Theta_1$ and $C_2[P_2] \xrightarrow{\mu} \Theta_2$, with $\Delta_i \, \mathtt{lift}(\mathcal{R}) \, \Theta_i$ and $C[Q] \xrightarrow{\tau} \Theta = \Theta_1 \mid \Theta_2$. It remains to prove that $\Delta \, \mathtt{lift}(\mathcal{R}) \, \Theta$. By $\Delta_i \, \mathtt{lift}(\mathcal{R}) \, \Theta_i$, there are index sets J, K such that $\Delta_1 = \sum_{j \in J} p_j \cdot \mathtt{dirac}(M_j)$ and $\Theta_1 = \sum_{j \in J} p_j \cdot \mathtt{dirac}(M'_j)$, and $\Delta_2 = \sum_{k \in K} q_k \cdot \mathtt{dirac}(N_k)$ and $\Theta_2 = \sum_{k \in K} q_k \cdot \mathtt{dirac}(N'_k)$, and for all j, k, $M_j \mathrel{\mathcal{R}} M'_j$ and $N_k \mathrel{\mathcal{R}} N'_k$. As a consequence, we have that $\Delta = \sum_{(j,k)} p_j \cdot q_k \cdot \mathtt{dirac}(M_j \mid N_k)$ and $\Theta = \sum_{(j,k)} p_j \cdot q_k \cdot \mathtt{dirac}(M'_j \mid N'_k)$, and it follows from $M_j \mathrel{\mathcal{R}} M'_j$ and $N_k \mathrel{\mathcal{R}} N'_k$ that $M_j \mid N_k \mathrel{\mathcal{R}} M'_j \mid N'_k$.

We now prove that if $C[P]$ and $C[Q]$ are probabilistic bisimilar with respect to actions τ and ω, then they have the same probability of success. There are two main differences with respect to the proof for the λ-calculus case in the probabilistic setting:

- $C[P]$ (and $C[Q]$) have internal nondeterminism with respect to τ-actions, hence we have different possible resolutions to consider.

- a state reached by $C[P]$ might perform both ω and τ; Hence, given a specific resolution $\mathcal{Z}$ of $C[P]$ it does not hold that the probability of success of $z_{C[P]}$ is $\sum_{n \geq 0} \mathit{prob}(z_{C[P]}, \tau^n \omega)$. Indeed, $\mathit{prob}(z_{C[P]}, \tau^n \omega)$ also includes the weights of paths where not only the final state, but also some states before, are successful, and such weights should not be added to the probability of success, since they do not correspond to weights of successful paths. Hence, we are going to consider probabilistic failure traces instead of probabilistic traces.

Since $P \sim_{\mathrm{PFTr}} Q$ on RPLTS implies that P, Q have the same initial labels, we have that P, Q are equivalent also with respect to "extended" probabilistic failure traces of the form $((F_i, r_i)_{i \leq n} F)$, where an initial failure set is allowed.

Let $\mathcal{SC}_n P$ denote the set of successful computations from P of length n. Then for any RPLTS P its probability of success at length n is $\mathit{prob}(\mathcal{SC}_n((P)) = \sum_{c \in \mathcal{SC}_n P} \mathit{prob}(c)$.

Hence, we have that $\mathit{prob}(\mathcal{SC}(P)) = \sum_n \mathit{prob}(\mathcal{SC}_n((P)) = \mathit{prob}(P, (\{\omega\}\tau)^n \emptyset \omega \emptyset)$.

We know from [BDL14a; BDL13] that on NPLTSs, as for LTSs, bisimilarity implies failure trace equivalence.

Hence, it follows from $C[P] \sim_{\mathrm{PB}} C[Q]$ with respect to labels τ, ω that $C[P] \sim_{\mathrm{PFTr}} C[Q]$ with respect to the same labels. Since $C[P]$ and $C[Q]$, this is equivalent to saying that for every resolutions $\mathcal{Z}_{C[P]}$ of $C[P]$ there is a resolution $\mathcal{Z}_{C[Q]}$ on $C[Q]$ such that $z_{C[P]} \sim_{\mathrm{PFTr}} z_{C[Q]}$ (where, since z_P and z_Q are RPLTSs, the last instance of $\sim_{\mathrm{PFTr}}$ is defined as on RPLTSs). Hence, for every $n \geq 0$, given the (extended) failure trace $(\{\omega\}\tau)^n \emptyset \omega \emptyset$ we have $\mathit{prob}(z_{C[P]}, (\{\omega\}\tau)^n \emptyset \omega \emptyset) = \mathit{prob}(z_{C[Q]}, (\{\omega\}\tau)^n \emptyset \omega \emptyset)$, and we derive that

$$\begin{aligned} \mathit{prob}(\mathcal{SC}(z_{C[P]})) &= \textstyle\sum_n \mathit{prob}(\mathcal{SC}_n((z_{C[P]})) \\ &= \textstyle\sum_n \mathit{prob}(z_{C[P]}, (\{\omega\}\tau)^n \emptyset \omega \emptyset) \\ &= \textstyle\sum_n \mathit{prob}(z_{C[Q]}, (\{\omega\}\tau)^n \emptyset \omega \emptyset) \\ &= \textstyle\sum_n \mathit{prob}(\mathcal{SC}_n((z_{C[Q]})) \\ &= \mathit{prob}(\mathcal{SC}(z_{C[Q]})) \end{aligned}$$

Symmetrically, for every resolution $\mathcal{Z}_{C[Q]}$ of $C[Q]$ there is a resolution $\mathcal{Z}_{C[P]}$ of $C[P]$ such that $prob(\mathcal{SC}(z_{C[Q]})) = prob(\mathcal{SC}(z_{C[P]}))$.

As a consequence,

$$\bigsqcup_{\mathcal{Z}_{C[P]} \in Res_{\tau,max}(C[P])} prob(\mathcal{SC}(z_{C[P]})) = \bigsqcup_{\mathcal{Z}_{C[Q]} \in Res_{\tau,max}(C[Q])} prob(\mathcal{SC}(z_{C[Q]}))$$

Chapter 4

Probabilistic testing

In the previous chapter we have considered testing equivalences induced by contexts of languages that may have higher-order and/or concurrent features, but that do not have probabilistic features. In this chapter, we study testing equivalences for RPLTSs where the tests may have also probabilistic choices. Instead of considering testing equivalences where tests are given by contexts from some language, we define the equivalences using tests that are semantically defined. In particular, we consider three different classes of observers respectively formalized as RPLTS, LTS, and NPLTS. These can be seen as the semantics of terms of first-order process calculi, possibly allowing probabilistic choices. In order to apply such a test to an RPLTS, we look at the interactions between the RPLTS and the observer running in parallel.

In Section 4.1 we introduce the testing scenario used in this chapter. We give upper and lower bounds to the discriminating power of the three classes of observers on RPLTSs in Section 4.2.1, and then we investigate the relationships among the resulting testing equivalences (Section 4.2.2). We conclude by discussing two open problems and conjectures (Section 4.3). Detailed proofs of the results presented in this chapter can be found in Section 4.4.

4.1 Testing equivalences for RPLTS processes

Given an RPLTS, we assume that the elements of its action set $\mathcal{A}$ are all visible. The action set of each considered test will be $\bar{\mathcal{A}} \cup \{\omega\}$, where $\bar{\mathcal{A}} = \{\bar{a} \mid a \in \mathcal{A}\}$ is the set of coactions for $\mathcal{A}$ and $\omega \notin \mathcal{A}$ is a distinguished action denoting success. Every coaction must synchronize with the corresponding action; when this happens, the invisible action $\tau \notin \mathcal{A}$ is produced. Therefore, the resulting interaction system is an NPLTS with action set $\{\tau, \omega\}$, whose transition relation $\longrightarrow$ is derived from the transition relation $\longrightarrow_1$ of the RPLTS process under test and the transition relation $\longrightarrow_2$ of the observer, through the following two rules:

$$\frac{s \xrightarrow{a}_1 \Delta_1 \quad o \xrightarrow{\bar{a}}_2 \Delta_2}{(s, o) \xrightarrow{\tau} (\Delta_1, \Delta_2)} \qquad \frac{o \xrightarrow{\omega}_2 \Delta_2}{(s, o) \xrightarrow{\omega} (\mathtt{dirac}(s), \Delta_2)}$$

where $(\Delta, \Theta)(s', o') = \Delta(s') \cdot \Theta(o')$. This operation corresponds to putting in parallel the tested process and the observer seen as terms of some language, and only considering the

τ-labeled transitions resulting from the synchronization of inputs from the tested term with outputs from the observer.

We then apply the definitions of computations, successful computations, and resolutions we have seen in Section 2.3 to the NPLTS resulting from the observer interacting with the RPLTS. This allows us to define testing equivalences analogously to the previous chapters. In contrast with previous chapters, we consider directly *test-equivalence*, which is the intersection of may- and must-equivalence. May equivalence is defined by considering the supremum of the probabilities of success of al resolutions. By contrast, must-equivalence checks whether the infimum is the same. The relationship between may- and must-equivalences in this setting is further discussed in Section 4.3.1.

Given a resolution $\mathcal{Z}$ of (s, o), we denote by $\mathcal{SC}(z_{s,o})$ the set of successful computations from the state $z_{s,o}$ of $\mathcal{Z}$ corresponding to (s, o). We respectively denote by $\sqcup$ and $\sqcap$ the supremum and the infimum of the set of probability values $prob(\mathcal{SC}(z_{s,o}))$ computed in the various resolutions of the interaction system. To avoid infima to be trivially zero, in the next definition, which is inspired by [YL92; JY95; KN98], we restrict ourselves to maximal resolutions.

Definition 4.1. Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow_{\mathcal{L}})$ be an RPLTS. We say that $s_1, s_2 \in S$ are *probabilistic* $\sqcup\sqcap$*-testing equivalent*, written $s_1 \sim_{\text{PTe-}\sqcup\sqcap} s_2$, iff for every test $\mathcal{T} = (O, \bar{\mathcal{A}}, \longrightarrow_{\mathcal{T}})$ with initial state $o \in O$ it holds that:

$$\bigsqcup_{\mathcal{Z}_1 \in Res_{\max}(s_1,o)} prob(\mathcal{SC}(z^{\mathcal{Z}_1}_{s_1,o})) = \bigsqcup_{\mathcal{Z}_2 \in Res_{\max}(s_2,o)} prob(\mathcal{SC}(z^{\mathcal{Z}_2}_{s_2,o}))$$
$$\bigsqcap_{\mathcal{Z}_1 \in Res_{\max}(s_1,o)} prob(\mathcal{SC}(z^{\mathcal{Z}_1}_{s_1,o})) = \bigsqcap_{\mathcal{Z}_2 \in Res_{\max}(s_2,o)} prob(\mathcal{SC}(z^{\mathcal{Z}_2}_{s_2,o}))$$

The equivalence is respectively denoted by $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$, $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$, or $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$ depending on whether the considered tests are all reactive probabilistic, (fully) nondeterministic, or nondeterministic and probabilistic.

We assume tests to be finite, i.e., finite state, finitely branching, and loop free. On the one hand, this entails that interaction systems will have finitely many maximal resolutions, thus ensuring the validity of our results also for a slightly finer variant of $\sim_{\text{PTe-}\sqcup\sqcap}$ that we could define following [Seg96; DGHM08]. On the other hand, this restriction will be exploited in the proofs of some results.

4.2 Properties of the RPLTS testing equivalences

4.2.1 Placing the testing equivalences in the RPLTS spectrum

Our first result is that the three relations $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$, $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$, and $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$ are comprised between $\sim_{\text{PFTr}}$ and $\sim_{\text{PB}}$. This confirms the power of the interplay between probabilities and nondeterminism for discriminating purposes, which was already noticed in the testing theory for NPLTS processes [JHSY94; DGHMZ07b; BDL14b].

The proof that each of the three equivalences is strictly finer than $\sim_{\text{PFTr}}$ benefits from an analogous result with respect to $\sim_{\text{PF}}$. Both proofs focus on tests that are deterministic LTS models (DLTS for short) as they admit neither internal nondeterminism nor probabilities. Since these tests constitute a submodel common to RPLTS, LTS, and NPLTS tests, the inclusion proofs relying on them scale to the three more expressive families of tests.

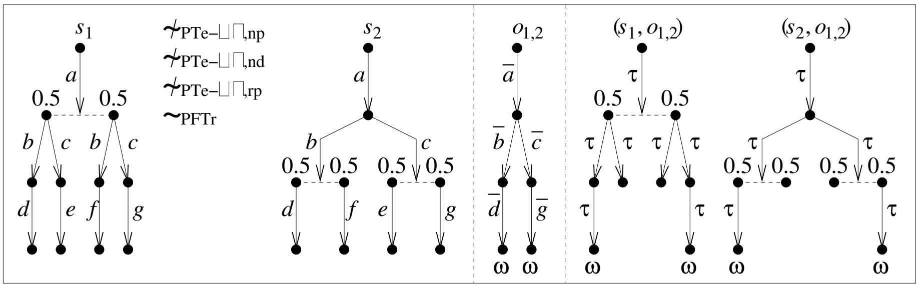
s_1
$\not\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$
$\not\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$
$\not\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$
$\sim_{\text{PFTr}}$
s_2
$o_{1,2}$
$(s_1, o_{1,2})$
$(s_2, o_{1,2})$
a
0.5
0.5
b
c
b
c
d
e
f
g
$\bar{a}$
$\bar{b}$
$\bar{c}$
$\bar{d}$
$\bar{g}$
τ
ω

Figure 4.1: Counterexample for testing equivalences and probabilistic failure trace equivalence on RPLTSs

Lemma 4.2. *On RPLTS processes, for all* $* \in \{\text{rp}, \text{nd}, \text{np}\}$ *it holds that:*

$$\sim_{\text{PTe-}\sqcup\sqcap,*} \subsetneq \sim_{\text{PF}}$$

Theorem 4.3. *On RPLTS processes, for all* $* \in \{\text{rp}, \text{nd}, \text{np}\}$ *it holds that:*

$$\sim_{\text{PTe-}\sqcup\sqcap,*} \subsetneq \sim_{\text{PFTr}}$$

The inclusions in $\sim_{\text{PFTr}}$ are strict as shown by the two RPLTS processes, the DLTS test, and the two NPLTS interaction systems in Figure 4.1, because we have $\sqcup = 1$ and $\sqcap = 0$ in the first system and $\sqcup = \sqcap = 0.5$ in the second one.

The proof that $\sim_{\text{PB}}$ is included in each of the three testing equivalences exploits the fact that $\sim_{\text{PB}}$ is a congruence with respect to parallel composition. Inclusion stems from showing that, under $\sim_{\text{PB}}$, for each maximal resolution of any of the two interaction systems, there exists a maximal resolution of the other interaction system, such that the two resolutions have the same success probability.

Theorem 4.4. *On RPLTS processes, for all* $* \in \{\text{rp}, \text{nd}, \text{np}\}$ *it holds that:*

$$\sim_{\text{PB}} \subseteq \sim_{\text{PTe-}\sqcup\sqcap,*}$$

4.2.2 Relationships among the RPLTS testing equivalences

Our second result is concerned with the relationships among the discriminating powers of $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$, $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$, and $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$, which will help us investigating the strictness of the inclusions of Theorem 4.4.

First of all, we observe that $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$ is included both in $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$ and in $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$, because RPLTS tests and LTS tests are special cases of NPLTS tests. Both inclusions are strict, as shown in the upper part of Figure 4.2, where the NPLTS test yields $\sqcup = 0.75$ and $\sqcap = 0.25$ in the first interaction system and $\sqcup = \sqcap = 0.5$ in the second one. We remark the need of both internal nondeterminism and probabilities in the distinguishing test. A linear test succeeding after performing $\bar{a}$, $\bar{b}$, and $\bar{c}$ would not be able to tell apart s_3 and s_4. Likewise, those two states would not be distinguishable by a test obtained from the previous one by replacing the $\bar{c}$-transition with a probabilistic choice between that transition and a terminal/success state, or introducing a nondeterministic choice through a further $\bar{b}$-transition to a terminal/success state after the $\bar{a}$-transition.

Secondly, it turns out that, in general, $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$ and $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$ are incomparable with each other. For instance, in the middle part of Figure 4.2 we have that $s_5 \sim_{\text{PTe-}\sqcup\sqcap,\text{rp}} s_6$, while $s_5 \not\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}} s_6$ because the LTS test yields $\sqcup = 1$ and $\sqcap = 0$ in the first

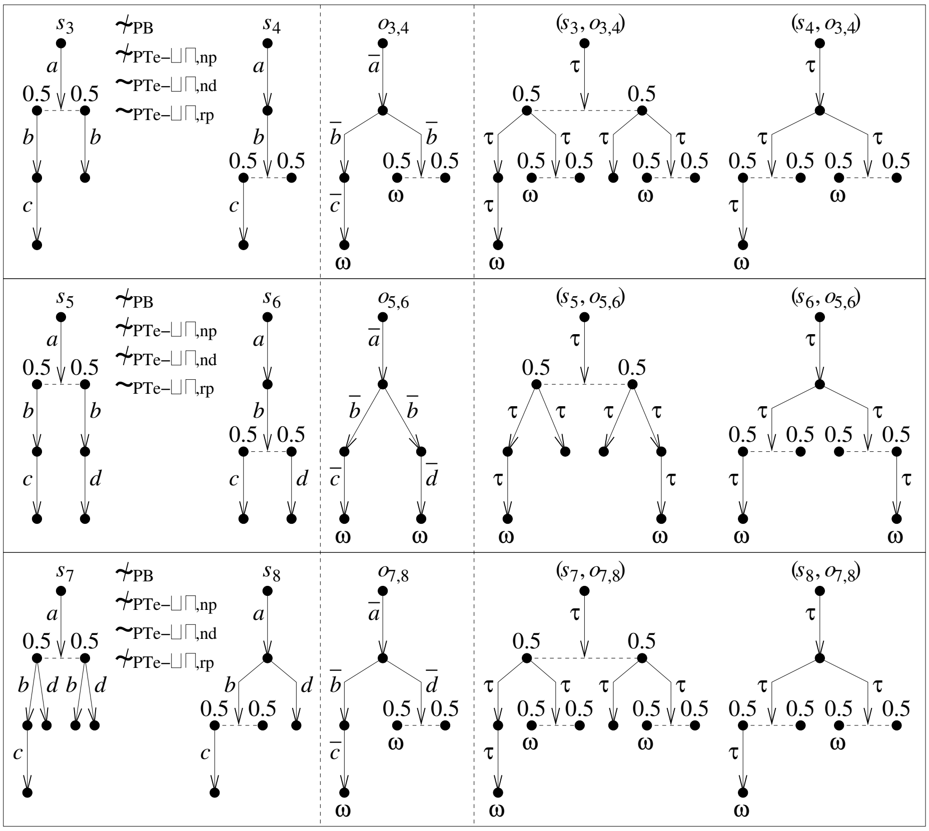
s_3
$\not\sim_{\rm PB}$
$\not\sim_{\rm PTe\text{-}\sqcup\sqcap,np}$
$\sim_{\rm PTe\text{-}\sqcup\sqcap,nd}$
$\sim_{\rm PTe\text{-}\sqcup\sqcap,rp}$
s_4
$o_{3,4}$
$(s_3, o_{3,4})$
$(s_4, o_{3,4})$
s_5
$\not\sim_{\rm PB}$
$\not\sim_{\rm PTe\text{-}\sqcup\sqcap,np}$
$\not\sim_{\rm PTe\text{-}\sqcup\sqcap,nd}$
$\sim_{\rm PTe\text{-}\sqcup\sqcap,rp}$
s_6
$o_{5,6}$
$(s_5, o_{5,6})$
$(s_6, o_{5,6})$
s_7
$\not\sim_{\rm PB}$
$\not\sim_{\rm PTe\text{-}\sqcup\sqcap,np}$
$\sim_{\rm PTe\text{-}\sqcup\sqcap,nd}$
$\not\sim_{\rm PTe\text{-}\sqcup\sqcap,rp}$
s_8
$o_{7,8}$
$(s_7, o_{7,8})$
$(s_8, o_{7,8})$
a
b
c
d
$\bar{a}$
$\bar{b}$
$\bar{c}$
$\bar{d}$
τ
0.5
ω

Figure 4.2: Counterexamples for probabilistic bisimilarity and testing equivalences on RPLTSs

interaction system and $\sqcup = \sqcap = 0.5$ in the second one. Notice the necessity of internal nondeterminism in the distinguishing test. In contrast, in the lower part of Figure 4.2 we have that $s_7 \sim_{\rm PTe\text{-}\sqcup\sqcap,nd} s_8$, while $s_7 \not\sim_{\rm PTe\text{-}\sqcup\sqcap,rp} s_8$ because the RPLTS test yields $\sqcup = 0.75$ and $\sqcap = 0.25$ in the first interaction system and $\sqcup = \sqcap = 0.5$ in the second one. Unlike the upper part of Figure 4.2, here internal nondeterminism is not necessary in the distinguishing test.

Thirdly, if $\sim_{\rm PTe\text{-}\sqcup\sqcap,rp}$ admitted only restricted RPLTS tests, then it would include $\sim_{\rm PTe\text{-}\sqcup\sqcap,nd}$, with the inclusion being strict as shown in the middle part of Figure 4.2. A restricted RPLTS (RRPLTS for short) test is a test such that its probabilistic choices, i.e., its non-Dirac transitions, are not preceded by nondeterministic choices. The proof of this fact is based on the deprobabilization of an RRPLTS test. This is an algorithm that performs a top-down traversal of the test until a set of DLTS subtests is generated, which preserves the extremal success probabilities induced by the original test.

When encountering a non-Dirac transition in the top-down traversal of the RRPLTS test, as shown in Figure 4.3 the algorithm replaces the test with as many RRPLTS subtests – which are DLTS subtests in the final steps – as there are ways of resolving the probabilistic choice. For simplicity, only the non-Dirac transition, labeled with $\bar{a}$, originating the probabilistic choice is depicted in the figure, but in general it could be the last transition in a computation – traversing states where no nondeterministic choices occur – going from

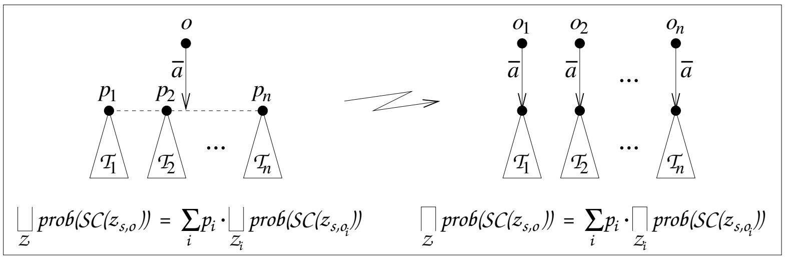
o
ā
p_1
p_2
p_n
$\mathcal{T}_1$
$\mathcal{T}_2$
$\mathcal{T}_n$
...
o_1
o_2
o_n
ā
ā
ā
$\mathcal{T}_1$
$\mathcal{T}_2$
$\mathcal{T}_n$
...
$\bigsqcup_{\mathcal{Z}} prob(SC(z_{s,o})) = \sum_i p_i \cdot \bigsqcup_{\mathcal{Z}_i} prob(SC(z_{s,o_i}))$
$\bigsqcap_{\mathcal{Z}} prob(SC(z_{s,o})) = \sum_i p_i \cdot \bigsqcap_{\mathcal{Z}_i} prob(SC(z_{s,o_i}))$

Figure 4.3: Deprobabilization of an RRPLTS test (applies recursively to $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$)

the initial state o of the test to the probabilistic choice. Given a state s of the process under test, the two formulas in Figure 4.3 witness that the two convex combinations of the extremal success probabilities induced by the n subtests respectively coincide with the two extremal success probabilities induced by the original test.

Should a nondeterministic choice precede the considered probabilistic choice, it would not be appropriate to generate subtests by resolving both choices. The reason is that it would then be natural to focus on the maximum and the minimum of the extremal success probabilities induced by the various subtests arising from the resolution of the nondeterministic choice. This certainly works when the nondeterministic choice is originated from the initial state of the test, or from the state reached by a Dirac transition of the test that synchronizes with a Dirac transition of the process under test. However, the synchronization of a Dirac transition of the test with a non-Dirac transition of the process results in a non-Dirac transition in the interaction system, for which a convex combination (as opposed to maximum and minimum) of the extremal success probabilities of the various subtests needs to be computed.

Fourthly, if $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$ admitted only DLTS tests, then it would include $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$, with the inclusion being strict as shown in the lower part of Figure 4.2. The reason is that a DLTS test is a special case of RPLTS test in which there are no probabilistic choices. In conclusion, we have:

Theorem 4.5. *On RPLTS processes, it holds that:*

1. $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}} \subsetneq \sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$ *and* $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}} \subsetneq \sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$.

2. $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$ *and* $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$ *are incomparable with each other.*

3. $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}} \subsetneq \sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$ *if only RRPLTS tests were admitted by* $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}}$.

4. $\sim_{\text{PTe-}\sqcup\sqcap,\text{rp}} \subsetneq \sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$ *if only DLTS tests were admitted by* $\sim_{\text{PTe-}\sqcup\sqcap,\text{nd}}$.

It follows from Theorem 4.4 and from Theorem 4.5(1) that the testing equivalence induced by RPLTS or LTS tests is strictly included in probabilistic bisimilarity.

Corollary 4.6. *On RPLTS processes, for all* $* \in \{\text{rp}, \text{nd}\}$ *it holds that:*

$$\sim_{\text{PB}} \subsetneq \sim_{\text{PTe-}\sqcup\sqcap,*}$$

4.3 Open problems and conjectures

4.3.1 May vs. must testing

In the case of testing LTS or NPLTS processes, it is known that must testing equivalence is strictly finer than may testing equivalence in the absence of divergence, otherwise the two equivalences are incomparable [Nic87; DGHMZ07b]. When testing RPLTS processes, the relationships between $\sim_{\text{PTe-}\sqcup}$ (may testing) and $\sim_{\text{PTe-}\sqcap}$ (must testing) are not clear, even if we restrict ourselves to NPLTS tests and we admit τ-actions within them.

In that case, we could derive that $\sim_{\text{PTe-}\sqcap,\text{np}} \subseteq \sim_{\text{PTe-}\sqcup,\text{np}}$ by exploiting the construction used in [DGHMZ07b] for proving an analogous result on NPLTS processes. The purpose of that construction is to build from a given NPLTS test a dual one, which generates all complementary success probabilities in the interaction system. The idea is to transform every state of the test having an outgoing ω-transition into a terminal state, and to add to any other state a τ-transition followed by an ω-transition.

The absence of internal nondeterminism within RPLTS processes would however prevent us from concluding that the above inclusion is strict. Indeed, the typical counterexample made out of a test succeeding after performing $\bar{a}$ followed by $\bar{b}$, which distinguishes a process that can perform either a followed by b, or a followed by c, from a process that can perform a and then has a choice between b and c, is not applicable because the first process is not an RPLTS.

Such considerations lead us to conjecture that, for each of the three variants of $\sim_{\text{PTe-}\sqcup\sqcap}$, its may part $\sim_{\text{PTe-}\sqcup}$ coincides with its must part $\sim_{\text{PTe-}\sqcap}$, and hence both coincide with $\sim_{\text{PTe-}\sqcup\sqcap}$ by virtue of the definition of the latter. This is certainly true when restricting attention to fully probabilistic tests – as they yield, when interacting with an RPLTS process, a single maximal resolution, in which $\sqcup$ and $\sqcap$ necessarily coincide – or tests having exactly one nondeterministic choice that occurs in the initial state – as can be easily proved by induction on the number of maximal resolutions of each such test.

4.3.2 Characterizing RPLTS testing equivalences

Our findings in Section 4.2 leave open the question whether $\sim_{\text{PB}}$ is strictly finer than $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$ or coincides with it. In the latter case, we would have that, in the RPLTS setting, testing equivalence reaches the same discriminating power as bisimilarity not only in the presence of an explicit copying capability within tests [LS91], but also in the absence of it, provided that tests are equipped with both internal nondeterminism and probabilities. We point out that this would be a peculiarity of RPLTS processes, because it is known that NPLTS tests are less powerful than bisimilarity in the case of NPLTS processes [BDL14a].

The numerous examples of RPLTS processes that we have examined lead us to conjecture that on RPLTS processes $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}} = \sim_{\text{PB}}$. As a consequence of Theorem 4.4, it suffices to prove that $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$ is included in $\sim_{\text{PB}}$. This is equivalent to showing that, given two states s_1 and s_2 of an RPLTS, if $s_1 \not\sim_{\text{PB}} s_2$, then $s_1 \not\sim_{\text{PTe-}\sqcup\sqcap,\text{np}} s_2$.

The idea is to use a logic characterizing probabilistic bisimilarity on RPLTS to build a distinguishing NPLTS test. Probabilistic Modal Logic (PML), a modal logic characterizing $\sim_{\text{PB}}$ on RPLTS, was first proposed in [LS91] and then led to a minimal form in [DEP02]. PML comprises the always true constant $\top$, logical conjunction $\cdot \land \cdot$, and the diamond operator $\langle a \rangle_p \cdot$ where a is an action and p is a probability lower bound. Formula $\langle a \rangle_p F$

is satisfied by an RPLTS state if an a-labeled transition is possible from that state, after which a set of states satisfying F is reached with probability at least p. The proof of the conjecture appears far from being trivial. The connection between PML and the testing approach of [LS91] is intuitively clear, as multiplying the success probabilities resulting from the application of independent choice-free tests to as many copies of the current state under test is analogous to taking the logical conjunction of a number of formulas each starting with a suitably decorated diamond. In contrast, the tests used in this chapter follow the classical theory of [DH84], hence do not admit any copying capability and, most importantly, may contain choices, which fit well together with logical disjunction rather than conjunction. The characterization of probabilistic bisimulation via a modal logic based on disjunction has been recently studied in [BM16], and could be used as basis for proving the conjecture.

We conclude by mentioning that an alternative proof strategy for this conjecture, when only considering may testing equivalence, may exploit Proposition 2.12 ($\sim_{\rm PB} = \sim_{\rm PS}$) and the characterization of may testing via simulation provided by [DGHM08]. However, we recall that in [DGHM08] τ-actions are admitted, the considered probabilistic simulation is not the standard one, and the focus is on preorders rather than equivalences.

4.4 Proofs

Proof of Lemma 4.2

We first prove that the same result holds for probabilistic trace equivalence.

Lemma 4.7. *On RPLTS processes, for all* $* \in \{\mathrm{rp}, \mathrm{nd}, \mathrm{np}\}$ *it holds that:*

$$\sim_{\text{PTe-}\sqcup\sqcap,*} \subsetneq \sim_{\rm PTr}$$

Proof. Since $\sim_{\text{PTe-}\sqcup\sqcap,*}$ is included in $\sim_{\text{PTe-}\sqcup,*}$, it is sufficient to prove that the latter is included in $\sim_{\rm PTr}$. Moreover, let us restrict ourselves to consider only DLTS tests, in which neither internal nondeterminism nor probabilities are allowed, and denote by $\sim_{\text{PTe-}\sqcup,\rm d}$ the may part of the resulting probabilistic testing equivalence. Since a DLTS is a submodel common to RPLTS, LTS, and NPLTS, $\sim_{\text{PTe-}\sqcup,*}$ is included in $\sim_{\text{PTe-}\sqcup,\rm d}$. Thus, if we prove the inclusion in $\sim_{\rm PTr}$ for the DLTS case, then the inclusion in $\sim_{\rm PTr}$ will hold also for the other three cases.
Given an RPLTS $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ and $s_1, s_2 \in S$, we consider the contrapositive statement. If $s_1 \not\sim_{\rm PTr} s_2$, i.e., if there exists a trace $\sigma \in \mathcal{A}^*$ such that $prob(\mathcal{C}(s_1, \sigma)) \neq prob(\mathcal{C}(s_2, \sigma))$, then the DLTS test $\mathcal{T}_\sigma$ with initial state o_σ having a single maximal computation that is labeled with $\bar{\sigma}\,\omega$ yields:

$$\begin{aligned}\bigsqcup_{\mathcal{Z}_1 \in Res_{\max}(s_1, o_\sigma)} prob(\mathcal{SC}(z_{s_1, o_\sigma})) \;=\; prob(\mathcal{C}(s_1, \sigma)) \;\neq\; \\ \neq\; prob(\mathcal{C}(s_2, \sigma)) \;=\; \bigsqcup_{\mathcal{Z}_2 \in Res_{\max}(s_2, o_\sigma)} prob(\mathcal{SC}(z_{s_2, o_\sigma}))\end{aligned}$$

which means that $s_1 \not\sim_{\text{PTe-}\sqcup,\rm d} s_2$. □

As in the previous proof, to show that all testing equivalences are included in $\sim_{\rm PF}$ it is sufficient to prove the inclusion of $\sim_{\text{PTe-}\sqcup,\rm d}$ in $\sim_{\rm PF}$.

Given an RPLTS $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ and $s_1, s_2 \in S$, suppose that $s_1 \sim_{\text{PTe-}\sqcup,\text{d}} s_2$. For an arbitrary failure pair $\varphi = (\sigma, F)$, where $F \neq \emptyset$ to avoid overlapping with $\sim_{\text{PTr}}$, we consider a DLTS test $\mathcal{T}_\varphi$ with initial state o_φ that can only perform a computation labeled with $\bar{\sigma}$, after which a state is reached having an outgoing $\bar{a}$-transition followed by an ω-transition for each $a \in F$.
For all $s \in S$ it holds that:

$$\bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s, o_\varphi)} \mathit{prob}(\mathcal{SC}(z_{s,o_\varphi})) \; = \; \mathit{prob}(\mathcal{C}(s, \sigma)) - \mathit{prob}(\mathcal{FC}(s, \varphi))$$

hence we have $s_1 \sim_{\text{PTe-}\sqcup,\text{d}} s_2$ and $s_1 \sim_{\text{PTr}} s_2$ (by Lemma 4.7), and it follows that:

$$\begin{aligned} \mathit{prob}(\mathcal{FC}(s_1, \varphi)) \; &= \; \mathit{prob}(\mathcal{C}(s_1, \sigma)) - \bigsqcup_{\mathcal{Z}_1 \in \mathit{Res}_{\max}(s_1, o_\varphi)} \mathit{prob}(\mathcal{SC}(z_{s_1,o_\varphi})) \; = \\ = \; \mathit{prob}(\mathcal{C}(s_2, \sigma)) &- \bigsqcup_{\mathcal{Z}_2 \in \mathit{Res}_{\max}(s_2, o_\varphi)} \mathit{prob}(\mathcal{SC}(z_{s_2,o_\varphi})) \; = \; \mathit{prob}(\mathcal{FC}(s_2, \varphi)) \end{aligned}$$

which means that $s_1 \sim_{\text{PF}} s_2$.

Proof of Theorem 4.3

As in the previous proof, it is sufficient to demonstrate the inclusion of $\sim_{\text{PTe-}\sqcup,\text{d}}$ in $\sim_{\text{PFTr}}$.
Given an RPLTS $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ and $s_1, s_2 \in S$, suppose that $s_1 \sim_{\text{PTe-}\sqcup,\text{d}} s_2$. For an arbitrary failure trace $\phi = (a_1, F_1)\,(a_2, F_2) \dots (a_n, F_n)$, where $n \geq 1$, $a_i \notin F_{i-1}$ for all $i = 2, \dots, n$, and $F_i \neq \emptyset$ for some $i = 1, \dots, n$ to avoid trivial cases as well as overlapping with $\sim_{\text{PTr}}$, we consider a DLTS test $\mathcal{T}_\phi$ with initial state o_ϕ that can only perform a computation labeled with $\bar{a}_1 \bar{a}_2 \dots \bar{a}_n$ such that, for all $i = 1, \dots, n$, the state reached after performing $\bar{a}_i$ has also an outgoing $\bar{a}$-transition followed by an ω-transition for each $a \in F_i$.

For all $s \in S$ it holds that:

$$\begin{aligned} \bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s, o_\phi)} \mathit{prob}(\mathcal{SC}(z_{s,o_\phi})) \; = \; & \mathit{prob}(\mathcal{C}(s, a_1)) \\ & - \sum_{i=2}^{n} \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1} \cup \{a_i\}))) \\ & - \mathit{prob}(\mathcal{FTC}(s, \phi)) \end{aligned}$$

The reason is that, for all $i = 1, \dots, n-1$, given a computation of s labeled with $a_1 \dots a_i$, for each state in the support of the target distribution reached after performing a_i there are the following three alternative cases:

- the state can perform at least one action in F_i, thereby leading to success in the interaction with $\mathcal{T}_\phi$;
- the state can perform neither actions in F_i nor action a_{i+1}, thereby leading to failure in the interaction with $\mathcal{T}_\phi$;
- the state can perform no actions in F_i but can perform action a_{i+1};

where the last two cases boil down to the same one leading to failure when $i = n$ (the state can perform no actions in F_n). Therefore:

$$\bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s, o_\phi)} \mathit{prob}(\mathcal{SC}(z_{s,o_\phi})) \; = \; \sum_{i=1}^{n} \mathit{prob}(\mathcal{C}'(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,(a_i, \exists F_i)))$$

where $\mathcal{C}'(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,(a_i, \exists F_i))$ is the set of computations of s compatible with the failure trace $(a_1, F_1) \dots (a_{i-1}, F_{i-1})$ that can subsequently perform action a_i and reach a state in which at least one action in F_i is enabled. For all $i = 1, \dots, n$ it holds that:

$$\begin{array}{l} \mathit{prob}(\mathcal{C}'(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,(a_i, \exists F_i))) \\ = \mathit{prob}(\mathcal{C}''(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,a_i)) \\ - \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,(a_i, F_i))) \end{array}$$

where $\mathcal{C}''(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,a_i)$ is the set of computations of s compatible with the failure trace $(a_1, F_1) \dots (a_{i-1}, F_{i-1})$ that can subsequently perform action a_i, hence $\mathcal{C}''(s, a_1) = \mathcal{C}(s, a_1)$ while for all $i = 2, \dots, n$ it holds that:

$$\begin{array}{rcl} \mathit{prob}(\mathcal{C}''(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,a_i)) & = & \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1}))) \\ & & - \, \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1} \cup \{a_i\}))) \end{array}$$

Summing up:

$$\begin{array}{rcl} \bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s, o_\phi)} \mathit{prob}(\mathcal{SC}(z_{s, o_\phi})) & = & [\mathit{prob}(\mathcal{C}(s, a_1)) - \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1)))] \\ & & + \sum_{i=2}^{n} [\mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1}))) \\ & & \quad - \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1} \cup \{a_i\}))) \\ & & \quad - \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1})\,(a_i, F_i)))] \\ & = & \mathit{prob}(\mathcal{C}(s, a_1)) \\ & & - \sum_{i=2}^{n} \mathit{prob}(\mathcal{FTC}(s, (a_1, F_1) \dots (a_{i-1}, F_{i-1} \cup \{a_i\}))) \\ & & - \, \mathit{prob}(\mathcal{FTC}(s, \phi)) \end{array}$$

Recall now that $s_1 \sim_{\text{PTe-}\sqcup,\text{d}} s_2$, hence $s_1 \sim_{\text{PTr}} s_2$. We show that $\mathit{prob}(\mathcal{FTC}(s_1, \phi)) = \mathit{prob}(\mathcal{FTC}(s_2, \phi))$ by proceeding by induction on $\mid \phi \mid= n \geq 1$:

- Let $n = 1$. Then:

$$\begin{array}{l} \mathit{prob}(\mathcal{FTC}(s_1, \phi)) \; = \; \mathit{prob}(\mathcal{C}(s_1, \sigma)) - \bigsqcup_{\mathcal{Z}_1 \in \mathit{Res}_{\max}(s_1, o_\phi)} \mathit{prob}(\mathcal{SC}(z_{s_1, o_\phi})) \; = \\ = \; \mathit{prob}(\mathcal{C}(s_2, \sigma)) - \bigsqcup_{\mathcal{Z}_2 \in \mathit{Res}_{\max}(s_2, o_\phi)} \mathit{prob}(\mathcal{SC}(z_{s_2, o_\phi})) \; = \; \mathit{prob}(\mathcal{FTC}(s_2, \phi)) \end{array}$$

- Let $n > 1$ and suppose that the result holds for each failure trace of length j such that $1 \leq j \leq n - 1$. Then:

$$\begin{array}{rcl} \mathit{prob}(\mathcal{FTC}(s_1, \phi)) & = & \mathit{prob}(\mathcal{C}(s_1, a_1)) \\ & & - \sum_{i=2}^{n} \mathit{prob}(\mathcal{FTC}(s_1, (a_1, F_1) \dots (a_{i-1}, F_{i-1} \cup \{a_i\}))) \\ & & - \bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s_1, o_\phi)} \mathit{prob}(\mathcal{SC}(z_{s_1, o_\phi})) \\ & = & \mathit{prob}(\mathcal{C}(s_2, a_1)) \\ & & - \sum_{i=2}^{n} \mathit{prob}(\mathcal{FTC}(s_2, (a_1, F_1) \dots (a_{i-1}, F_{i-1} \cup \{a_i\}))) \\ & & - \bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s_2, o_\phi)} \mathit{prob}(\mathcal{SC}(z_{s_2, o_\phi})) \\ & = & \mathit{prob}(\mathcal{FTC}(s_2, \phi)) \end{array}$$

We can thus conclude that $s_1 \sim_{\mathrm{PFTr}} s_2$.

Proof of Theorem 4.4

We prove that $\sim_{\mathrm{PB}}$ implies $\sim_{\mathrm{PTe}\text{-}\sqcup\sqcap,\mathrm{np}}$.

We know that $\sim_{\mathrm{PB}}$ on NPLTS is preserved by the parallel operator [SL95], which implies that if $(s_1, s_2) \in \sim_{\mathrm{PB}}$ then $((s_1, o), (s_2, o)) \in \sim_{\mathrm{PB}}$, for o an arbitrary NPLTS test. Then the proof proceeds analogously to the proof of Theorem 3.13 (item (1)). Bisimilarity implies failure trace equivalence on NPLTS [BDL13], which means that for each resolution $\mathcal{Z}_1$ of (s_1, o) there is a resolution $\mathcal{Z}_2$ of (s_2, o) such that for every failure trace computation ϕ:

$$\mathit{prob}(\mathcal{FTC}(z_{s_1,o}, \phi)) \;=\; \mathit{prob}(\mathcal{FTC}(z_{s_2,o}, \phi))$$

and vice versa.

Let $\mathcal{SC}_n(z_{s_i,o})$ denote the set of successful computations from $z_{s_i,o}$ of length n, for $i \in \{1,2\}$. Then the probability of success at length n of $z_{s_i,o}$ is $\mathit{prob}(\mathcal{SC}_n(z_{s_i,o})) = \sum_{c \in \mathcal{SC}_n(z_{s_i},o)} \mathit{prob}(c)$, and it is in turn equal to $\mathit{prob}(\mathcal{FTC}(z_{s_i,o}, (\{\omega\}\tau)^n \emptyset\omega\emptyset))$.

Hence, we derive

$$\mathit{prob}(\mathcal{SC}(z_{s_i,o})) = \sum_n \mathit{prob}(\mathcal{SC}_n(z_{s_i,o})) = \sum_n \mathit{prob}(\mathcal{FTC}(z_{s_i,o}, (\{\omega\}\tau)^n \emptyset\omega\emptyset))\,.$$

(Note that we are here allowing a failure set also at the beginning of the failure trace. As we have seen, failure trace equivalence allows us to do so.)

Since for each resolution $\mathcal{Z}_1$ of (s_1, o) there is a resolution $\mathcal{Z}_2$ of (s_2, o) such that for every n,

$$\mathit{prob}(\mathcal{FTC}(z_{s_1,o}, (\{\omega\}\tau)^n \emptyset\omega\emptyset)) \;=\; \mathit{prob}(\mathcal{FTC}(z_{s_2,o}, (\{\omega\}\tau)^n \emptyset\omega\emptyset))$$

(and vice versa) the result follows.

Proof of Theorem 4.5

Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an RPLTS:

1. The two inclusions immediately follow from the fact that LTS tests and RPLTS tests are special cases of NPLTS tests.

2. Incomparability stems from the middle part and the lower part of Figure 4.2.

3. First of all, we establish the correctness of the deprobabilization algorithm for RRPLTS tests, i.e., the fact that the set of DLTS subtests generated by the algorithm preserves the extremal success probabilities induced by the original RRPLTS test. More precisely, given $s \in S$ and an RRPLTS test $\mathcal{T} = (O, \mathcal{A}, \longrightarrow_{\mathcal{T}})$ with initial state $o \in O$, it holds that:

$$\bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) \;=\; \sum_{j=1}^{k} q_j \cdot \bigsqcup_{\mathcal{Z}'_j \in \mathit{Res}_{\max}(s,o'_j)} \mathit{prob}(\mathcal{SC}(z_{s,o'_j}))$$

$$\bigsqcap_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) \;=\; \sum_{j=1}^{k} q_j \cdot \bigsqcap_{\mathcal{Z}'_j \in \mathit{Res}_{\max}(s,o'_j)} \mathit{prob}(\mathcal{SC}(z_{s,o'_j}))$$

as we prove below by proceeding by induction on the number $k \in \mathbb{N}_{\geq 1}$ of DLTS subtests $\mathcal{ST}'_1, \ldots, \mathcal{ST}'_k$ with initial states $o'_1, \ldots, o'_k$ and associated probabilities $q_1, \ldots, q_k$ generated for $\mathcal{T}$ by the deprobabilization algorithm:

- If $k = 1$, then $\mathcal{T}$ has no non-Dirac transitions at all, and hence the only DLTS test $\mathcal{ST}'_1$ with initial state $o'_1 = o$ and associated probability 1 generated by the deprobabilization algorithm coincides with $\mathcal{T}$. In this case, the result trivially holds.

- Let $k \geq 2$ and assume that the result holds for all RRPLTS tests for which the deprobabilization algorithm generates at most $k - 1$ DLTS subtests. From $k \geq 2$, it follows that $\mathcal{T}$ has at least one non-Dirac transition. Consider the first of these transitions encountered in the top-down traversal of $\mathcal{T}$, whose target distribution is supposed to assign to the states in its support the probability values p_i, $1 \leq i \leq n$, with $n \in \mathbb{N}_{\geq 2}$. Let $\mathcal{ST}_i$, $1 \leq i \leq n$, be the corresponding RRPLTS subtests generated by the deprobabilization algorithm, with initial states o_i for $i = 1, \dots, n$ (see Figure 4.3).
Due to the absence in $\mathcal{T}$ of nondeterministic choices preceding the considered non-Dirac transition, we have that:

$$\begin{aligned}\bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) &= \sum_{i=1}^{n} p_i \cdot \bigsqcup_{\mathcal{Z}_i \in \mathit{Res}_{\max}(s,o_i)} \mathit{prob}(\mathcal{SC}(z_{s,o_i})) \\ \bigsqcap_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) &= \sum_{i=1}^{n} p_i \cdot \bigsqcap_{\mathcal{Z}_i \in \mathit{Res}_{\max}(s,o_i)} \mathit{prob}(\mathcal{SC}(z_{s,o_i}))\end{aligned}$$

Since the application of the deprobabilization algorithm to each such subtest $\mathcal{ST}_i$ generates $k_i \leq k - 1$ DLTS subtests (which are DLTS subtests of $\mathcal{T}$ too) $\mathcal{ST}'_{i,h}$, $1 \leq h \leq k_i$, with initial states $o'_{i,1}, \dots, o'_{i,k_i}$ and associated probabilities $q_{i,1}, \dots, q_{i,k_i}$, by the induction hypothesis we derive that:

$$\begin{aligned}\bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) &= \sum_{i=1}^{n} p_i \cdot \sum_{h=1}^{k_i} q_{i,h} \cdot \bigsqcup_{\mathcal{Z}'_{i,h} \in \mathit{Res}_{\max}(s,o'_{i,h})} \mathit{prob}(\mathcal{SC}(z_{s,o'_{i,h}})) \\ \bigsqcap_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) &= \sum_{i=1}^{n} p_i \cdot \sum_{h=1}^{k_i} q_{i,h} \cdot \bigsqcap_{\mathcal{Z}'_{i,h} \in \mathit{Res}_{\max}(s,o'_{i,h})} \mathit{prob}(\mathcal{SC}(z_{s,o'_{i,h}}))\end{aligned}$$

which can be rewritten as follows due to the distributivity of multiplication with respect to addition:

$$\begin{aligned}\bigsqcup_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) &= \sum_{i=1}^{n} \sum_{h=1}^{k_i} (p_i \cdot q_{i,h}) \cdot \bigsqcup_{\mathcal{Z}'_{i,h} \in \mathit{Res}_{\max}(s,o'_{i,h})} \mathit{prob}(\mathcal{SC}(z_{s,o'_{i,h}})) \\ \bigsqcap_{\mathcal{Z} \in \mathit{Res}_{\max}(s,o)} \mathit{prob}(\mathcal{SC}(z_{s,o})) &= \sum_{i=1}^{n} \sum_{h=1}^{k_i} (p_i \cdot q_{i,h}) \cdot \bigsqcap_{\mathcal{Z}'_{i,h} \in \mathit{Res}_{\max}(s,o'_{i,h})} \mathit{prob}(\mathcal{SC}(z_{s,o'_{i,h}}))\end{aligned}$$

Given $s_1, s_2 \in S$, suppose now that $s_1 \sim_{\text{PTe-}\sqcup\sqcap,\text{nd}} s_2$ and consider an arbitrary RRPLTS test $\mathcal{T} = (O, \mathcal{A}, \longrightarrow_{\mathcal{T}})$ with initial state $o \in O$ for which the deprobabilization algorithm generates $k \in \mathbb{N}_{\geq 1}$ DLTS subtests $\mathcal{ST}'_1, \dots, \mathcal{ST}'_k$ with initial states $o'_1, \dots, o'_k$ and associated probabilities $q_1, \dots, q_k$. From $s_1 \sim_{\text{PTe-}\sqcup\sqcap,\text{nd}} s_2$, it follows in particular that s_1 and s_2 cannot be told apart by any DLTS test, hence:

$$\begin{aligned}
\bigsqcup_{\mathcal{Z}_1 \in \mathit{Res}_{\max}(s_1,o)} \mathit{prob}(\mathcal{SC}(z_{s_1,o})) &= \sum_{j=1}^{k} q_j \cdot \bigsqcup_{\mathcal{Z}'_{1,j} \in \mathit{Res}_{\max}(s_1,o'_j)} \mathit{prob}(\mathcal{SC}(z_{s_1,o'_j})) \\
&= \sum_{j=1}^{k} q_j \cdot \bigsqcup_{\mathcal{Z}'_{2,j} \in \mathit{Res}_{\max}(s_2,o'_j)} \mathit{prob}(\mathcal{SC}(z_{s_2,o'_j})) \\
&= \bigsqcup_{\mathcal{Z}_2 \in \mathit{Res}_{\max}(s_2,o)} \mathit{prob}(\mathcal{SC}(z_{s_2,o})) \\
\bigsqcap_{\mathcal{Z}_1 \in \mathit{Res}_{\max}(s_1,o)} \mathit{prob}(\mathcal{SC}(z_{s_1,o})) &= \sum_{j=1}^{k} q_j \cdot \bigsqcap_{\mathcal{Z}'_{1,j} \in \mathit{Res}_{\max}(s_1,o'_j)} \mathit{prob}(\mathcal{SC}(z_{s_1,o'_j})) \\
&= \sum_{j=1}^{k} q_j \cdot \bigsqcap_{\mathcal{Z}'_{2,j} \in \mathit{Res}_{\max}(s_2,o'_j)} \mathit{prob}(\mathcal{SC}(z_{s_2,o'_j})) \\
&= \bigsqcap_{\mathcal{Z}_2 \in \mathit{Res}_{\max}(s_2,o)} \mathit{prob}(\mathcal{SC}(z_{s_2,o}))
\end{aligned}$$

4. The inclusion immediately follows from the fact that DLTS tests are special cases of RPLTS tests.

Chapter 5

Conclusions

5.1 Additional related works

There are analogies between our results on the contextual equivalences induced by higher-order languages on ordinary LTSs and results in the literature on the equivalences on LTSs that characterize the coarsest congruences contained in trace equivalence for operators whose operational rules comply with certain *rule formats*. Some of these formats allow negative premises in the rules, with which refusals may be encoded, or allow rules in which an argument of an operator may end up, in the derivative of the rule, within a predefined context; when the context is polyadic, this yields a form of copying. In higher-order languages, in contrast, copying is achieved through the variable binding mechanisms of the languages. Passivation or, in the λ-calculi, call-by-value, are necessary to obtain the discriminating power of powerful formats such as GSOS [BIM95] and tyft/tyxt [GV92] (which give ready simulation equivalence and simulation equivalence, respectively).

Rule formats for probabilistic processes include [Bar02; LT09; DL12], where the emphasis is on ensuring congruence properties for bisimilarity. Testing of reactive probabilistic processes is studied in [KN98], obtaining an equivalence strictly coarser than bisimilarity, though the comparison with the equivalences induced by our contextual equivalences is unclear.

5.2 Conclusions and future work

In Chapter 3 we have studied the discriminating power offered by *higher-order concurrent* languages such as HOπ, without and with passivation, and contrasted it with those offered by *higher-order sequential* languages à la λ-calculus and by *first-order concurrent* languages à la CCS. We have measured this discriminating power on the basis of the distinctions that the languages, possibly extended with refusal, allow us to make on first-order processes that are either fully nondeterministic (LTSs) or reactive probabilistic (RPLTSs).

The discriminating power of HOπ with passivation coincides with that of the call-by-value λ-calculus, both on LTSs and on RPLTSs. Intuitively, HOπ with passivation and the call-by-value λ-calculus are both capable of implementing the 'and' of two tests. That is, the equivalence induced by these languages are characterized by modal logics that include the 'and' connective between formulas and are therefore 'branching-sensitive'.

$$\begin{tikzcd}
& \begin{array}{c}\text{ready simulation equivalence}\\ \Lambda_{\mathrm{V}}^{loc} = \mathrm{HO}\pi_{\mathrm{ref},\mathtt{pass}}\end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c}\text{simulation equivalence}\\ \Lambda_{\mathrm{V-ref}}^{loc} = \mathrm{HO}\pi_{\mathtt{pass}}\end{array} \arrow[dr] & & \begin{array}{c}\text{failure trace equivalence}\\ \Lambda_{\mathrm{N}}^{loc} = \mathrm{HO}\pi_{\mathrm{ref}} = \mathrm{CCS}^{-}_{\mathrm{ref}}\end{array} \arrow[dl] \\
& \begin{array}{c}\Lambda_{\mathrm{N-ref}}^{loc} = \mathrm{HO}\pi = \mathrm{CCS}^{-}\\ \text{trace equivalence}\end{array} &
\end{tikzcd}$$

Figure 5.1: The spectrum of equivalences for nondeterministic processes

The addition of refusal increases the discriminating power of all the considered languages when testing LTSs. This is not always the case on RPLTSs. One reason is that, similarly to fully probabilistic processes [JS90; JL91], the spectrum of equivalences for RPLTSs is narrower than for LTSs.

On LTSs, the extra discriminating power offered in concurrency by passivation over higher-order communication corresponds, in λ-calculi, to the call-by-value possibility of reducing the argument of a function and then capturing the result.

On RPLTSs, we do not know exactly what are the equivalences induced by $\mathrm{CCS}^{-}$and $\mathrm{CCS}^{-}_{\mathrm{ref}}$, though we know they are strictly in between probabilistic failure-trace equivalence and probabilistic bisimilarity. We are not aware of RPLTS equivalences in the literature with the same property. The lack of any copying facility makes the CCS^{-} equivalence also strictly coarser than those of $\mathrm{HO}\pi$ and of all other concurrent languages considered. Another question that remains open is whether the equivalences induced by $\mathrm{HO}\pi$ (with or without refusal) coincide, and whether they are strictly coarser than probabilistic bisimilarity.

Figures 5.1 and 5.2 summarize the relationship among the various equivalences on a first-order LTS or RPLTS, respectively, that have been considered in Chapter 3. In the figures, the name of a language, say AL, stands for the contextual equivalence $\simeq^{\mathcal{L}}_{\mathtt{AL}}$. A single arrow denotes a strict inclusion, unless the arrow is coupled with a question mark, in which case we do not know whether the inclusion is strict or not.

The contextual equivalences we have focused on in Chapter 3 are 'may' forms of contextual equivalence. We have discussed a few instances of 'must' contextual equivalence, and we leave it as future work to systematically address the must-equivalences.

When testing LTSs and RPLTSs with contexts from CCS-like or higher-order languages in Chapter 3, we have admitted probabilities in the tested first-order processes, but not in the testing languages. It would be interesting to see if and how the addition of probabilities to the testing languages affects the results. A first attempt at answering this question is given by the testing scenarios presented in Chapter 4, where RPLTS processes are tested using probabilistic tests as well. We have considered testing equivalences induced on RPLTSs by three classes of tests: LTS-like tests, RPLTS-like tests and NPLTS-like tests. The testing equivalence induced by RPLTS and LTS tests are incomparable with each other and they both are strictly more discriminating than probabilistic failure trace

$$\begin{tikzcd}
 & \begin{array}{c}\text{probabilistic bisimilarity}\\ \Lambda^{loc}_{\mathrm{V-ref}} = \Lambda^{loc}_{\mathrm{V}} = \mathrm{HO}\pi_{\mathtt{pass}} = \mathrm{HO}\pi_{\mathrm{ref},\mathtt{pass}}\end{array} \arrow[d, "?"] & \\
 & \mathrm{HO}\pi_{\mathrm{ref}} \arrow[dl, "?"'] \arrow[dr] & \\
\mathrm{HO}\pi \arrow[dr] & & \mathrm{CCS}^{-}_{\mathrm{ref}} \arrow[dl] \\
 & \mathrm{CCS}^{-} \arrow[d] & \\
 & \text{probabilistic failure trace equivalence } \Lambda^{loc}_{\mathrm{N}} \arrow[d] & \\
 & \begin{array}{c}\Lambda^{loc}_{\mathrm{N-ref}}\\ \text{probabilistic trace equivalence}\end{array} &
\end{tikzcd}$$

Figure 5.2: The spectrum of equivalences for reactive probabilistic processes

equivalence on RPLTS processes. NPLTS tests induce a testing equivalence that is strictly more discriminating than the ones induced by RPLTS or LTS tests. In particular, in a language-based testing akin to that of Chapter 3, the case $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$ when the tests are NPLTS would correspond to testing RPLTS by putting them in contexts of the form $C = [\cdot] \mid M$, where M is a term from a probabilistic, finitary and synchronization-free CCS language. Hence, the discriminating power of the language is deeply increased by the presence of probabilities already when a restricted, first-order class of contexts is considered. Indeed, we have shown that probabilistic bisimilarity is included in $\sim_{\text{PTe-}\sqcup\sqcap,\text{np}}$ and we have conjectured that these equivalences actually coincide. This would mean that probabilistic bisimilarity can be captured by first-order tests (and languages) featuring both probability and nondeterminism.

The tested LTSs/RPLTSs processes in this work do not feature internal (i.e., τ-labeled) moves, which means that the induced equivalences are 'strong'. A natural developments of this work thus consists in admitting internal actions in the tested processes and therefore move to 'weak' behavioral relations.

Finally, we have examined the equivalences induced on purely nondeterministic processes (LTSs), and on reactive probabilistic processes (RPLTSs), but we have not considered combinations of them; this would amount to studying whether the contextual equivalences induced on NPLTSs coincide with known probabilistic testing equivalences, e.g., [Seg96; DGHM08; DGHM09] (characterized also as variants of simulation), or other behavioral relations are needed. The study of the spectrum of equivalences for NPLTSs is a non-trivial extension of the one for RPLTSs, since many of the characterization results for probabilistic equivalences presented in the previous chapters rely on the fact that the considered processes only feature external nondeterminism. For instance, the proof that the contextual equivalences induced by languages $\Lambda^{loc}_{\mathrm{V-ref}}, \Lambda^{loc}_{\mathrm{V}}$, HO$\pi_{\mathtt{pass}}$ and HO$\pi_{\mathrm{ref},\mathtt{pass}}$ all collapse and coincide with probabilistic bisimilarity on RPLTSs (Theorems 3.9 and 3.13) could not be adapted to the case when the tested processes are NPLTSs. The proof that these contextual equivalences imply probabilistic bisimilarity exploits the peculiar result that probabilistic bisimilarity and probabilistic similarity coincide on RPLTSs, and that on RPLTS they are captured by tests that only admit conjunction and testing of actions (see Section 2.3.3).

Part II

Full abstraction for probabilistic λ-calculi

Chapter 6

Background

We recall the definitions of applicative and environmental bisimulations for pure (non-probabilistic) λ-calculi, and we discuss counterexamples motivating the use of environmental bisimulations for richer languages such as imperative λ-calculi.[4] Then in Section 6.2 we introduce the semantics of pure probabilistic call-by-name and call-by-value λ-calculi.

6.1 Bisimulations for λ-calculi

In pure non-probabilistic λ-calculi (Section 2.4), the definition of contextual equivalence is based on observing convergence, or termination. We say that a term M converges (notation: $M \Downarrow$) if $M \Longrightarrow V$ for some value V. Otherwise, we say that M diverges (notation: $M \Uparrow$).

Definition 6.1 ([Mor68])**.** Terms M, N of the call-by-name λ-calculus (respectively: call-by-value) are contextually equivalent if for every context C of the calculus, $C[M] \Downarrow$ iff $C[N] \Downarrow$.

Example 6.2. Terms $\Omega = (\lambda x.xx)\,(\lambda x.xx)$ and $\mathsf{I} = \lambda x.x$ are respectively a diverging term and the identity function. Trivially, they are not contextually equivalent since using the empty context $C = [\cdot]$ we have $C[\Omega] \Uparrow$ and $C[\mathsf{I}] \Downarrow$. This holds both in call-by-name and in call-by-value.

Examples of contextually equivalent terms, both in call-by-name and in call-by-value, are the identity function I and its variant $\lambda x.\mathsf{I}\,x$. We will prove that they are contextually equivalent in call-by-value using applicative bisimulations in the following section.
Notation: We use M, N, L, P, Q for terms of λ-calculi, and V, W for values.

6.1.1 Applicative bisimulation

We define both applicative and environmental bisimulation using big-step clauses. This aims at simplifying the comparison between the non-probabilistic definitions and the definitions for probabilistic calculi, which we will present in the following chapters and which, as we will see, have to be big-step.

[4] The results presented in this section and references to these results in the literature have been discussed in Section 1.2.3.

In applicative bisimulation, we test that equivalent terms both converge (clauses (1) and (2)) and, if so, that the reached values are equivalent whenever they are given the same argument (clause (3)).

Definition 6.3 (Applicative bisimulation, call-by-name [Abr90])**.** A relation $\mathcal{R}$ over closed terms of the call-by-name λ-calculus is an applicative bisimulation if $M \mathrel{\mathcal{R}} N$ implies that:

1. if $M \Longrightarrow V$ then $N \Longrightarrow W$ and $V \mathrel{\mathcal{R}} W$;
2. if $N \Longrightarrow W$ then $M \Longrightarrow V$ and $V \mathrel{\mathcal{R}} W$ (i.e., the symmetric condition, from N);
3. if $M = \lambda x.M'$ and $N = \lambda x.N'$ then for every term P, $M'\{{}^{P}\!/\!{}_{x}\} \mathrel{\mathcal{R}} N'\{{}^{P}\!/\!{}_{x}\}$.

Terms M and N are applicative bisimilar if there is an applicative bisimulation $\mathcal{R}$ such that $M \mathrel{\mathcal{R}} N$.

Definition 6.4 (Applicative bisimulation, call-by-value)**.** The definition for the call-by-value λ-calculus is the same except for the third clause, which becomes:

3′. if $M = \lambda x.M'$ and $N = \lambda x.N'$ then for every value V, $M'\{{}^{V}\!/\!{}_{x}\} \mathrel{\mathcal{R}} N'\{{}^{V}\!/\!{}_{x}\}$.

If we used general terms instead of values to test abstractions in call-by-value, then the terms $\lambda.\mathsf{I}$ (recall that a thunk $\lambda.M$ is a term $\lambda x.M$ where x does not occur free in M) and $\lambda x.(\lambda.\mathsf{I})x$ would not be bisimilar in call-by-value, since when x is substituted with argument Ω the first term converges and the second one diverges. However, the two terms are contextually equivalent in call-by-value.

The greatest applicative bisimulation (*applicative bisimilarity*) is an equivalence relation, and coincides with the union of all bisimulations. In pure, deterministic calculi, applicative bisimilarity coincides with applicative simulation equivalence. Simulations are defined, as usual, by removing the second (symmetric) clause in the definition of bisimulation.

The definition of applicative bisimulation can be recovered as the standard (first-order) bisimulation applied to an LTS, as defined in Section 2.2.2 (Definition 2.6). To this end, we define an LTS whose states are λ-terms and with transitions representing both the evaluation of λ-terms and the test of a value carried out by providing it with an argument.

We define here the LTS for the call-by-name λ-calculus.[5] Let Λ and $\mathcal{V}$ be the sets of closed terms and values of the calculus, respectively.

Definition 6.5. The LTS $(S, \mathcal{A}, \rightarrow)$ is given by:

- A set of states $S = \{\Lambda\} \uplus \{\hat{\mathcal{V}}\}$, where terms and values are taken modulo α-equivalence and $\hat{\mathcal{V}} = \{\hat{V} \mid V \in \mathcal{V}\}$ is a set containing copies of the values in Λ decorated with $\hat{}$. We call these values *distinguished values.*
- A set of labels $\mathcal{A} = \Lambda \cup \{\mathit{eval}\}$, where, again, terms are taken modulo α-equivalence.
- A transition relation $\rightarrow$ such that:
  - for every $M \in \Lambda$ and for every $\hat{V} \in \hat{\mathcal{V}}$, $M \xrightarrow{\mathit{eval}} \hat{V}$ iff $M \Longrightarrow V$;
  - for every $\widehat{\lambda x.M} \in \hat{\mathcal{V}}$ and for every $P \in \Lambda$, $\widehat{\lambda x.M} \xrightarrow{P} M\{{}^{P}\!/\!{}_{x}\}$.

[5] This LTS is built analogously to the ones in [DLSA14; CD14] for the probabilistic case, that we will present in the next chapter.

If $V \in \mathcal{V}$, then both V and $\hat{V}$ are states of the LTS. Distinguished values allow us to distinguish between a value V seen as term that is going to be reduced and a value $\hat{V}$ seen as the result of the reduction of a term.

The LTS for call-by-value is defined analogously, but with set of actions $\mathcal{A} = \mathcal{V} \cup \{eval\}$, since only values are given as arguments to functions in the bisimulation game. Hence, the last item of the definition of the transition relation becomes:

$$\text{for every } \widehat{\lambda x.M} \in \hat{\mathcal{V}} \text{ and for every } V \in \mathcal{V},\ \widehat{\lambda x.M} \xrightarrow{V} M\{V/x\}$$

Applicative bisimulations enjoy a simple and easy to apply definition, and have been proved to be fully abstract with respect to contextual equivalence both in pure call-by-name and in pure call-by-value λ-calculi (see [Pit12] and Section 1.2.3). Full abstraction means that applicative bisimilarity is sound (applicative bisimilarity implies contextual equivalence) and complete (contextual equivalence implies applicative bisimilarity), and thus coincides with contextual equivalence.

Example 6.6. By the full abstraction results, if we want to prove that in the call-by-value λ-calculus terms I and $\lambda x.\mathsf{I}\,x$ are contextually equivalent, we can exhibit the following relation:

$$\mathcal{R} = \{(\mathsf{I}, \lambda x.\mathsf{I}\,x)\} \cup \{(V, \mathsf{I}\,V) \mid V \text{ is a value}\} \cup \{(M, M) \mid M \text{ is a a term}\}$$

The relation is an applicative bisimulation since terms I and $\lambda x.\mathsf{I}\,x$ are already values and thereby trivially satisfy clauses (1) and (2). Then, whatever value V they are given as input we obtain a pair of the form $(V, \mathsf{I}\,V)$, which is in $\mathcal{R}$ by the second set and so clause (3) holds. In a pair of the form $(V, \mathsf{I}\,V)$ the two terms evaluate to the same value V, and we stay in the relation since identity is included in the relation. Finally, equal terms evaluate to equal values, and if we substitute the same terms to the same values we obtain equal terms. Hence, also the second and third sets in the definition of $\mathcal{R}$ satisfy the applicative bisimulation clauses.

6.1.2 Applicative vs. environmental bisimulation

We recall here the definition of environmental bisimulation [SKS11][6] and we discuss how it solves some drawbacks of applicative bisimilarity.

An *environmental relation* is a set of elements each of which is of the form $(\mathcal{E}, M, N)$ or $\mathcal{E}$, where M, N are closed terms and $\mathcal{E}$ is a relation on closed values. In a triple $(\mathcal{E}, M, N)$ the relation component $\mathcal{E}$ is the *environment*, and M, N are the *tested terms.* We write $M \mathcal{R}_{\mathcal{E}} N$ for $(\mathcal{E}, M, N) \in \mathcal{R}$. The contextual closure $\mathcal{R}^\star$ of a binary relation $\mathcal{R}$ is the set

$$\{(C[M_1, ..., M_n], C[N_1, ..., N_n]) \mid M_i \,\mathcal{R}\, N_i\}$$

Definition 6.7 (Environmental bisimulation, call-by-name)**.** An environmental relation $\mathcal{R}$ is an *environmental bisimulation* if

1. $M \mathcal{R}_{\mathcal{E}} N$ implies:

   (a) if $M \Longrightarrow V$ then $N \Longrightarrow W$ and $\mathcal{E} \cup \{(V, W)\} \in \mathcal{R}$;

[6] The definition of environmental bisimulation in [SKS11] uses small-step clauses. For uniformity with the rest of the thesis, we define here the big-step version.

   (b) the symmetric condition, from N;

2. if $\mathcal{E} \in \mathcal{R}$ then for all $(\lambda x.P, \lambda x.Q) \in \mathcal{E}$ and for all $(M, N) \in \mathcal{E}^\star$ it holds that $P\{^{M}\!/\!_{x}\}\mathcal{R}_{\mathcal{E}}Q\{^{N}\!/\!_{x}\}$.

Environmental bisimilarity is the union of all environmental bisimulations.

Hence, in environmental bisimulation terms are related with respect to a set of values $\mathcal{E}$ that grows during the bisimulation game. As in applicative bisimulation, related terms should both converge (clause (1)). In contrast with applicative bisimulation, the reached values are collected in environment $\mathcal{E}$, and the values in $\mathcal{E}$ are then tested by using arguments in the contextual closure of the same environment $\mathcal{E}$ (clause (2)). As a consequence, we are allowing a larger class of tests for functions with respect to applicative bisimulation (since the identity relation on terms is included in $\mathcal{E}^\star$).

Analogously to applicative bisimilarity, the definition for call-by-value is obtained by only allowing values as arguments for related abstractions. Formally, we write $\mathcal{R}^{\widehat{\star}}$ for the contextual closure of $\mathcal{R}$ restricted to values, and clause 2 becomes:

2′. if $\mathcal{E} \in \mathcal{R}$ then for all $(\lambda x.P, \lambda x.Q) \in \mathcal{E}$ and for all $(V, W) \in \mathcal{E}^{\widehat{\star}}$ it holds that $P\{^{V}\!/\!_{x}\}\ \mathcal{R}_{\mathcal{E}}\ Q\{^{W}\!/\!_{x}\}$.

Remark 6.8. In definition 6.7, we have defined the environment $\mathcal{E}$ as a set of pairs of values, following [SKS11]. This environment can be alternatively formalized using tuples of values. Instead of sets $\mathcal{E}$ we use pairs of tuples of values $(\widetilde{V}, \widetilde{W})$ where $\widetilde{V}$ and $\widetilde{W}$ have the same length, and then we define an environmental relation as a relation on pairs of the form either $(\widetilde{V}; M, \widetilde{W}; N)$, i.e., configurations of tuples of values each with a running term, or $(\widetilde{V}, \widetilde{W})$ (note that the relation is directly defined on tuples, so $\widetilde{V}\ \mathcal{R}\ \widetilde{W}$ does not denote the pointwise relation on corresponding values in the tuples). Then an environmental relation $\mathcal{R}$ on such pairs is an environmental bisimulation (for call-by-name) if

1. $\widetilde{V}; M\ \mathcal{R}\ \widetilde{W}; N$ implies:
   (a) if $M \Longrightarrow V$ then $N \Longrightarrow W$ and $\widetilde{V}, V\ \mathcal{R}\ \widetilde{W}, W$;
   (b) the symmetric condition, from N;
2. if $\widetilde{V}\ \mathcal{R}\ \widetilde{W}$ then for all $(\lambda x.P, \lambda x.Q)$ such that $(\widetilde{V})_i = \lambda x.P$ and $(\widetilde{W})_i = \lambda x.Q$ for some i (i.e., the i-th projections of the tuples) and for every $(M, N) \in (\widetilde{V}, \widetilde{W})^\star$ (i.e., $M = C[\widetilde{V}]$ and $N = C[\widetilde{W}]$ for some C) we have $\widetilde{V}; P\{^{M}\!/\!_{x}\}\ \mathcal{R}\ \widetilde{W}; Q\{^{N}\!/\!_{x}\}$.

The usefulness of this alternative definition will become clear in Chapter 8.

As argued in the introduction (Section 1.2.3), the definition of applicative bisimulation has some drawbacks, that can be solved by resorting to bisimulations with a more complex definition such as environmental bisimulations. First, it is generally hard to prove that applicative bisimilarity is a congruence. To prove congruence in a direct manner, we can try to show that the contextual closure of an applicative bisimulation $\mathcal{R}$ is itself an applicative bisimulation. Such a proof fails since we have to show that the application of values in $\mathcal{R}$ to pairs of terms in the contextual closure of the relation is again in the contextual closure of the relation. The problem is that when $\lambda x.M\,\mathcal{R}\,\lambda x.N$:

- we want to derive that $M\{{}^{P}\!/\!_{x}\}\ \mathcal{R}\ N\{{}^{Q}\!/\!_{x}\}$, for all (P, Q) in the contextual closure of $\mathcal{R}$;
- by the definition of applicative bisimulation, we can only derive that $M\{{}^{P}\!/\!_{x}\}\ \mathcal{R}\ N\{{}^{P}\!/\!_{x}\}$, for every P.

Hence, the congruence proof of applicative bisimulation is carried out using Howe's method [How89], which is based on building a syntax-based relation which enjoys substitutivity properties by definition. Then we have to prove that this relation coincides with applicative bisimilarity, by showing complicated properties of the defined relation. By contrast, the congruence proof for environmental bisimulation can be carried out directly, since by definition functions are tested with inputs built from the contextual closure of terms in the environment.

Secondly, applicative bisimilarity is not sound in many extensions of pure λ-calculi [KLS11]. In particular, suppose we add imperative features, namely higher-order references (with private locations), to the call-by-value calculus, along the lines of the languages in [KW06b; SKS11].

Reduction is now defined on configurations $\langle s\,;\,M\rangle$, where s is a store (a function mapping locations to values) and M is a term. We assume a countable set of locations l. The syntax of the calculus is extended with constructs for reading and writing in the store, for creating fresh locations, and for performing operations on constants (e.g., arithmetical operations or identity checks on integers), that include booleans, integers and the unit value $\star$. We only consider a minimal version of the language, that is however sufficient for our purposes in this section. We will consider an extended, probabilistic version of the calculus in Chapter 8.

The syntax of terms and values is:

$$
\begin{array}{rll}
M ::= & x & \text{variables} \\
\mid & c & \text{constants} \\
\mid & \lambda x.M & \text{functions} \\
\mid & M_1 M_2 & \text{applications} \\
\mid & (\boldsymbol{\nu}\, x := M_1) M_2 & \text{new location} \\
\mid & !l & \text{dereferencing} \\
\mid & l := M_2 & \text{assignments} \\
\mid & \mathtt{op}(M_1, ..., M_n) & \text{primitive operations} \\
\mid & \mathtt{if}\ M_1\ \mathtt{then}\ M_2\ \mathtt{else}\ M_3 & \text{if-then-else}
\end{array}
$$

$$V ::= c \mid \lambda x.M$$

The small-step reduction and the evaluation contexts are defined in Figure 6.1. We write $s[l \to V]$ to denote the the update of s (possibly an extension of s if l is not in the domain of s). The language is typed, to ensure that in any store update $s[l \to V]$, value V has the type appropriate for l. In all semantic rules, any configuration $\langle s\,;\,M\rangle$ is *well-formed*, in that M is closed and all the locations in M and s are in the domain of s.

In λ-calculi with imperative features, and in particular in calculi with a local store (where fresh locations might be created at run time, and not made available to the con-

$$\textsc{Beta}\ \frac{}{\langle s\,;\,(\lambda x.M)V\rangle \longrightarrow \langle s\,;\,M\{^{V}\!/\!_{x}\}\rangle}$$

$$\textsc{New}\ \frac{l\ \text{ not in the domain of } s}{\langle s\,;\,(\boldsymbol{\nu}\,x :=V)M\rangle \longrightarrow \langle s[l \to V]\,;\,M\{^{l}\!/\!_{x}\}\rangle}$$

$$\textsc{Assign}\ \frac{}{\langle s\,;\,l := V\rangle \longrightarrow \langle s[l \to V]\,;\,\star\rangle} \qquad \textsc{Deref}\ \frac{s(l) = V}{\langle s\,;\,!l\rangle \longrightarrow \langle s\,;\,V\rangle}$$

$$\textsc{IfTrue}\ \frac{}{\langle s\,;\ \texttt{if true then}\ M_1\ \texttt{else}\ M_2\rangle \longrightarrow \langle s\,;\,M_1\rangle}$$

$$\textsc{IfFalse}\ \frac{}{\langle s\,;\ \texttt{if false then}\ M_1\ \texttt{else}\ M_2\rangle \longrightarrow \langle s\,;\,M_2\rangle}$$

$$\textsc{PrimOp}\ \frac{\texttt{Prim}(\texttt{op}, \widetilde{c}) = c'}{\langle s\,;\,\texttt{op}(\widetilde{c})\rangle \longrightarrow \langle s\,;\,c'\rangle}$$

$$\textsc{Eval}\ \frac{\langle s\,;\,M\rangle \longrightarrow \langle s'\,;\,M'\rangle \quad C \text{ is an evaluation context}}{\langle s\,;\,C[M]\rangle \longrightarrow \langle s'\,;\,C[M']\rangle}$$

$$\begin{array}{lrl}\text{Evaluation contexts} & C := & [\cdot] \ \big|\ CM \ \big|\ VC \ \big|\ \texttt{if}\ C\ \texttt{then}\ M_1\ \texttt{else}\ M_2 \\ & & \big|\ \texttt{op}(\widetilde{c}, C, \widetilde{M}) \ \big|\ l := C\end{array}$$

Figure 6.1: Single-step reduction relation for imperative λ-calculus

texts), applicative bisimilarity is not sound. For instance, consider the following terms:

$$M \overset{\mathsf{def}}{=} (\boldsymbol{\nu}\,x :=0)(\lambda.\ \texttt{if}\ \ !x = 0\ \ \texttt{then}\ \ (x := 1\,\underline{\texttt{seq}}\,\texttt{true})\ \ \texttt{else}\ \ \Omega)$$

$$N \overset{\mathsf{def}}{=} \lambda.\texttt{true}$$

where $M_1\,\underline{\texttt{seq}}\,M_2$ denotes term $(\lambda.M_2)M_1$, i.e., the execution of M_1 and M_2 in sequence. The terms are not contextually equivalent, since they are discriminated by context $C = (\lambda x.(x\star)\,\underline{\texttt{seq}}\,(x\star))[\cdot]$, starting from the empty store $s = \emptyset$. When put in context C, a term is evaluated in argument position, then the produced value is copied two times and the two copies are executed in sequence. Term M creates a fresh location l that is set to 0, and evaluates to $V \overset{\mathsf{def}}{=} \lambda.\ \texttt{if}\ \ !l = 0\ \ \texttt{then}\ \ (l := 1\,\underline{\texttt{seq}}\,\texttt{true})\ \ \texttt{else}\ \ \Omega$. The first time the context applies V to $\star$, it converges to $\texttt{true}$ and sets the location to 1. Then, the second time $V\star$ is evaluated, it diverges. By contrast, $N\star$ always converges. As a consequence, we have $\langle\emptyset\,;\,C[M]\rangle \Uparrow$ and $\langle\emptyset\,;\,C[N]\rangle \Downarrow$.

Suppose we extend the definition of applicative bisimulation to the imperative calculus defined in this section, by simply considering terms evaluated with respect to a store. Using this definition, terms $\langle\emptyset\,;\,M\rangle$ and $\langle\emptyset\,;\,N\rangle$ would be applicative bisimilar by relation:

$$\begin{aligned}\mathcal{R} = \{&(\langle\emptyset\,;\,M\rangle, \langle\emptyset\,;\,N\rangle), (\langle l=0\,;\,\lambda.\,\mathtt{if}\ \ !l=0\ \ \mathtt{then}\ \ (l:=1\,\underline{\mathtt{seq}}\,\mathtt{true})\ \ \mathtt{else}\ \ \Omega\rangle, \langle\emptyset\,;\,N\rangle),\\ &(\langle l=1\,;\,\mathtt{true}\rangle, \langle\emptyset\,;\,\mathtt{true}\rangle)\}\end{aligned}$$

and so applicative bisimilarity would be unsound with respect to contextual equivalence. The reason for this is that bisimulations are forgetful, since they do not allow us to accumulate terms and possibly test them twice in a row.

The example above is inspired by [KLS11]. The same paper presents several interesting and more involved examples justifying the different features of the definition of environmental bisimulation and its increased complexity with respect to applicative ones.
Environmental bisimulations allow us to accumulate and reuse values in the environment, and would thereby be able to discriminate terms M and N. Indeed, environmental bisimulations are fully abstract with respect to contextual equivalence in imperative λ-calculi [SKS11].

6.2 Probabilistic λ-calculi

We extend the syntax of the pure λ-calculus with a binary choice operator $\oplus$, that we will interpret as a probabilistic, fair choice.
The terms of the probabilistic λ-calculus $\Lambda_\oplus$ are generated by the following grammar:

$$M, N ::= x \;\Big|\; \lambda x.M \;\Big|\; MN \;\Big|\; M \oplus N$$

The *values* are the terms of the form $\lambda x.M$ (the abstractions). We call $\mathcal{V}_\oplus$ the set of values. As usual, contexts are terms with holes $[\cdot]$.

6.2.1 Semantics

Because of the probabilistic nature of choice in $\Lambda_\oplus$, a program does not evaluate to a value, but rather evaluates to a probability subdistribution on values. Therefore, we need the following notions to define an evaluation relation.[7]

A *value (sub)distribution* is a function $\Delta : \mathcal{V}_\oplus \to [0,1]$, such that $\sum_{V\in\mathcal{V}_\oplus}\Delta(V) \le 1$. As we will see, we use subdistributions instead of distributions (i.e., we allow the total weight of a distribution to be strictly less than 1) in order to model divergence. We generally omit the prefix and use "distributions" to denote subdistributions, unless otherwise specified.
Given a value distribution Δ, we let $\mathrm{supp}(\Delta)$ denote the set of those values V such that $\Delta(V) > 0$. Given a set X of values, $\Delta(X)$ is the sum of the probabilities assigned to every element of X, i.e., $\Delta(X) = \sum_{V\in X}\Delta(V)$. Moreover, we define $\mathtt{weight}(\Delta) = \sum_V \Delta(V)$, which corresponds to the total weight of the distribution Δ. A value distribution Δ is finite whenever $\mathrm{supp}(\Delta)$ has finite cardinality. If V is a value, we write $\mathtt{dirac}(V)$ for the value distribution Δ such that $\Delta(W) = 1$ if $W = V$ and $\Delta(V) = 0$ otherwise. We use $\Delta \le \Theta$ for the pointwise preorder on value distributions and we let $\Delta, \Theta, \Phi, \Xi$ range over value distributions.

[7] We recall here and adapt to the present setting some notions concerning probability distributions introduced in Chapter 2.

As is [DZ12], we first define an *approximation* semantics, which attributes *finite* probability distributions to terms, and only later define the actual semantics, which is the least upper bound of all distributions obtained through the approximation semantics. The reason why the semantics is infinitary is that, intuitively, while in pure λ-calculi a term converges (i.e., it terminates its computation) in a finite number of small-step reductions, in probabilistic calculi there may be several terminating computation paths, in which the total number of reductions need not be finitary (see Example 6.11).

We define the approximation semantics in a big-step style, by means of a binary relation $\Downarrow$ between closed terms and value distributions, which is defined by the set of rules from Figure 6.2. Hence, $M \Downarrow \Delta$ means that Δ is an approximation of the semantics of M. As in non-probabilistic calculi, the application rule is different depending on the evaluation strategy. In call-by-name we use rule App-cbn, and in call-by-value we use rule App-cbv. A small-step semantics (of approximations) for probabilistic call-by-name and call-by-value calculi can be defined as in [DZ12]. In Chapter 8 we will exploit an alternative, small-step semantics for probabilistic λ-calculi.

$$\textsc{Empty}\ \frac{}{M \Downarrow \emptyset} \qquad \textsc{Value}\ \frac{}{V \Downarrow \mathtt{dirac}(V)} \qquad \textsc{Sum}\ \frac{M \Downarrow \Delta \qquad N \Downarrow \Theta}{M \oplus N \Downarrow \frac{1}{2}\cdot\Delta + \frac{1}{2}\cdot\Theta}$$

$$\textsc{App-cbn}\ \frac{M \Downarrow \Delta \qquad \{P\{{}^{N}\!/\!{}_{x}\} \Downarrow \Theta_P\}_{\lambda x.P \in \mathrm{supp}(\Delta)}}{MN \Downarrow \sum_{\lambda x.P \in \mathrm{supp}(\Delta)} \Delta(\lambda x.P)\cdot\Theta_P}$$

$$\textsc{App-cbv}\ \frac{M \Downarrow \Delta \qquad N \Downarrow \Phi \qquad \{P\{{}^{V}\!/\!{}_{x}\} \Downarrow \Theta_{P,V}\}_{\lambda x.P \in \mathrm{supp}(\Delta),\, V \in \mathrm{supp}(\Phi)}}{MN \Downarrow \sum_{V \in \mathrm{supp}(\Phi)} \Phi(V)\cdot(\sum_{\lambda x.P \in \mathrm{supp}(\Delta)} \Delta(\lambda x.P)\cdot\Theta_{P,V})}$$

Figure 6.2: Operational semantics for pure probabilistic λ-calculi

Definition 6.9. For any closed term M, we define the (infinitary) *big-steps semantics* $[\![M]\!]$ of M as $[\![M]\!] = \sup\{\Delta \mid M \Downarrow \Delta\}$.

Since distributions form an ω-complete partial order, and for every M the set of those distributions Δ such that $M \Downarrow \Delta$ is a countable directed set, the semantics is well-defined [DZ12].

Example 6.10. The semantics $[\![\Omega]\!]$ of the always diverging term $\Omega = (\lambda x.xx)\,(\lambda x.xx)$ is the distribution $\emptyset$ assigning probability 0 to every value. The semantics of $\mathsf{I} = \lambda x.x$ is the distribution $1 \cdot \mathtt{dirac}(\mathsf{I})$. In between, one can find terms such as $\mathsf{I} \oplus \Omega$, and $\mathsf{I} \oplus (\mathsf{I} \oplus \Omega)$, whose semantics are the probability distributions assigning $\frac{1}{2}$ and $\frac{3}{4}$ to I, repectively.

The following example shows why we adopt an infinitary semantics in probabilistic λ-calculi.

Example 6.11. Consider the terms

$$P \overset{\mathtt{def}}{=} RR \quad \text{and} \quad Q \overset{\mathtt{def}}{=} \lambda.\Omega\,, \quad \text{for} \quad R \overset{\mathtt{def}}{=} \lambda x.(xx) \oplus Q$$

Both P and Q have probability 1 of becoming term Q i.e., they have the same semantics $1 \cdot \mathtt{dirac}(Q)$. Intuitively, after some computation steps, P may become Q or may become P again, with equal probability. However, the semantics of P is given as the supremum of an infinite set of distributions Δ such that $P \Downarrow \Delta$, and none of these approximants coincides with $[\![P]\!] = 1 \cdot \mathtt{dirac}(Q)$, since Δ can only be a distribution assigning to Q a probability value strictly smaller than one.

In the following chapters, the pure (probabilistic) λ-calculi will be untyped, whereas we will find types convenient to treat the extension with store presented in Chapter 8.

6.2.2 Contextual preorder and equivalence

In contextual equivalence for probabilistic calculi, the observation $M \Downarrow$ becomes probabilistic. Instead of checking the possibility of convergence, we check the probability of convergence, i.e., $\mathtt{weight}(M)$. Then, a term M is contextually equivalent to N if for any context C, the probability of convergence of $C[M]$ is the same as to the probability of convergence of the program obtained by replacing M by N in C. In the contextual preorder, we require the probability of convergence of $C[M]$ to be less than or equal to that of $C[N]$.

Definition 6.12. Terms M, N are in the *contextual preorder* ($M \leq_{\mathtt{ctx}} N$) if for every context C of $\Lambda_\oplus$ such that $C[M]$ and $C[N]$ are closed terms, it holds that $\mathtt{weight}([\![C[M]]\!]) \leq \mathtt{weight}([\![C[N]]\!])$. M, N are *contextually equivalent* ($M =_{\mathtt{ctx}} N$) if $M \leq_{\mathtt{ctx}} N$, and $N \leq_{\mathtt{ctx}} M$.

Equivalently, $M =_{\mathtt{ctx}} N$ if for every context C such that $C[M]$ and $C[N]$ are closed terms, $\mathtt{weight}([\![C[M]]\!]) = \mathtt{weight}([\![C[N]]\!])$.

It is easy to verify that the contextual preorder is indeed a preorder, and analogously for equivalence. The definitions of contextual preorder and equivalence can be applied to both closed and open terms. If the term is open, the contexts can bind the free variables of terms.

Chapter 7

Full abstraction for probabilistic applicative simulation

In [DLSA14], Abramsky's applicative bisimulation [Abr90] is generalized to the call-by-name, untyped λ-calculus with a binary, fair, probabilistic choice [DZ12]. Probabilistic applicative bisimulation is shown to be a congruence, thus included in context equivalence. Completeness, however, fails, but can be recovered if call-by-value evaluation is considered, as shown in [CD14]. This can appear surprising, given that in nondeterministic λ-calculi, both when call-by-name *and* call-by-value evaluation are considered, applicative bisimilarity is a congruence, but *finer* than context equivalence [Las98]. But there is another, even less expected result: the aforementioned correspondence does not hold anymore if we consider applicative *simulation* and the contextual *preorder*.

The reason why this happens can be understood if one looks at the testing-based characterization of probabilistic similarity and bisimilarity from the literature [DEP02; BMOW05]: the class of tests characterizing *bi*similarity (see Section 2.3.3) is simple enough to allow any test to be implementable by a program context. This is impossible for tests characterizing similarity, which, as we will see in Section 7.4, include not only conjunction (which can be implemented as copying) but also disjunction, an operator that seems to require the underlying language to be parallel.

In this chapter we show that, indeed, the presence of Plotkin's parallel disjunction [Plo77; AO93] turns applicative similarity into a relation which coincides with the context preorder. This is done by checking that the proof of precongruence for applicative bisimilarity [DLSA14; CD14] continues to hold (Section 7.3), and by showing how tests involving conjunction and disjunction can be implemented by contexts (Section 7.4.1). This somehow completes the picture about how applicative (bi)similarity behaves in a probabilistic scenario.

7.1 Probabilistic applicative simulation and bisimulation

In this section we recall the notions of probabilistic applicative simulation and bisimulation from [DLSA14; CD14] for the pure, probabilistic λ-calculus presented in Section 6.2. We directly define the relations for call-by-value calculi. The definitions for call-by-name calculi can be obtained as usual by considering terms as arguments, instead of values.

Given a relation $\mathcal{R} \subseteq X \times Y$ and a set $Z \subseteq X$, let $\mathcal{R}(Z) = \{y \mid \exists x \in Z$ such that $x \mathcal{R} y\}$.

Definition 7.1. A relation $\mathcal{R} \subseteq \Lambda_\oplus \times \Lambda_\oplus$ is a probabilistic applicative simulation if $M \mathcal{R} N$ implies:

- for all $X \subseteq \mathcal{V}_\oplus$, $[\![M]\!](X) \leq [\![N]\!](\mathcal{R}(X))$
- if $M = \lambda x.M'$ and $N = \lambda x.N'$ then $M'\{V/x\} \mathcal{R} N'\{V/x\}$ for all $V \in \mathcal{V}_\oplus$.

A relation $\mathcal{R}$ is a probabilistic applicative bisimulation if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are probabilistic applicative simulations. We say that M is simulated by N ($M \lesssim N$) if there exists a probabilistic applicative simulation $\mathcal{R}$ such that $M \mathcal{R} N$. Terms M, N are bisimilar ($M \approx N$) if there exists a probabilistic applicative bisimulation $\mathcal{R}$ such that $M \mathcal{R} N$.

Analogously to what happens in pure λ-calculi, these definitions correspond to simulations and bisimulations on a probabilistic first-order system.
We show how to define an RPLTS representing terms of $\Lambda_\oplus$ and their evaluation. States in the RPLTS correspond to λ-terms, and the states M, N in the RPLTS are in the simulation preorder (respectively, bisimilar) if and only if terms M, N are in the applicative simulation preorder (respectively: applicative bisimilar).

In order to model divergence, we are now considering terms that evaluate to subdistributions. Hence, we loosen the definition of RPLTS by allowing subdistributions (as opposed to distributions) to be reached after performing a state transition, following [DEP02]. In what follows, we use RPLTS to denote systems defined as in Definition 2.3, except that we have subdistributions instead of distributions.

Definition 7.2. The Reactive Probabilistic Labeled Transition System $\mathcal{L}_\oplus = (S, \mathcal{A}, \longrightarrow)$ is given by:

- A set of states $S = \{\Lambda_\oplus\} \uplus \{\hat{\mathcal{V}}_\oplus\}$, where terms and values are taken modulo α-equivalence and $\hat{\mathcal{V}}_\oplus = \{\hat{V} \mid V \in \mathcal{V}_\oplus\}$ is the set of *distinguished values*, containing copies of the values in $\Lambda_\oplus$ decorated with $\hat{}$.
- A set of labels $\mathcal{A} = \mathcal{V}_\oplus \uplus \{eval\}$, where, again, terms are taken modulo α-equivalence.
- A probabilistic transition relation $\rightarrow \subseteq (S \times \mathcal{A} \times \mathcal{D}(S))$ such that:
  - for every $M \in \Lambda_\oplus$, $M \xrightarrow{eval} [\![\hat{M}]\!]$, with $[\![\hat{M}]\!]$ a probability subdistribution that behaves analogously to $[\![M]\!]$ on distinguished values, i.e., $[\![\hat{M}]\!](\hat{V}) = [\![M]\!](V)$ for every $\hat{V} \in \hat{\mathcal{V}}_\oplus$, and $[\![\hat{M}]\!](M') = 0$ for all $M' \in \Lambda_\oplus$;
  - for every $\widehat{\lambda x.M} \in \hat{\mathcal{V}}_\oplus$ and for every $V \in \mathcal{V}_\oplus$, $\widehat{\lambda x.M} \xrightarrow{V} \mathtt{dirac}(M\{V/x\})$.

If $V \in \mathcal{V}_\oplus$, then both V and $\hat{V}$ are states of the RPLTS $\mathcal{L}_\oplus$. This RPLTS is defined in [DLSA14] for the call-by-name untyped probabilistic λ-calculus $\Lambda_\oplus$, and for a call-by-value typed probabilistic version of PCF in [CD14]. As in the first-order LTS for non-probabilistic calculi (Definition 6.5), actions in $\mathcal{V}_\oplus$ and action *eval* respectively represent the application of a term to a value and the evaluation of a term.

On an RPLTS with subdistributions, instead of distributions, we can directly apply the definition of bisimulation as defined by Larsen and Skou in [LS91] (see definition (PB1) in Section 2.3).[8] For probabilistic simulation, we can use a definition based on the

[8] To apply the definition based on $\mathtt{lift}(\mathcal{R})$, we have to redefine the probabilistic lifting by allowing distributions to have weight smaller than 1. By contrast, when it comes to probabilistic simulation, requiring the reached probability subdistributions to be in the lifting $\mathtt{lift}(\mathcal{R})$ of the simulation relation $\mathcal{R}$ is too strong a condition, since it would imply that the subdistributions must have the same weight.

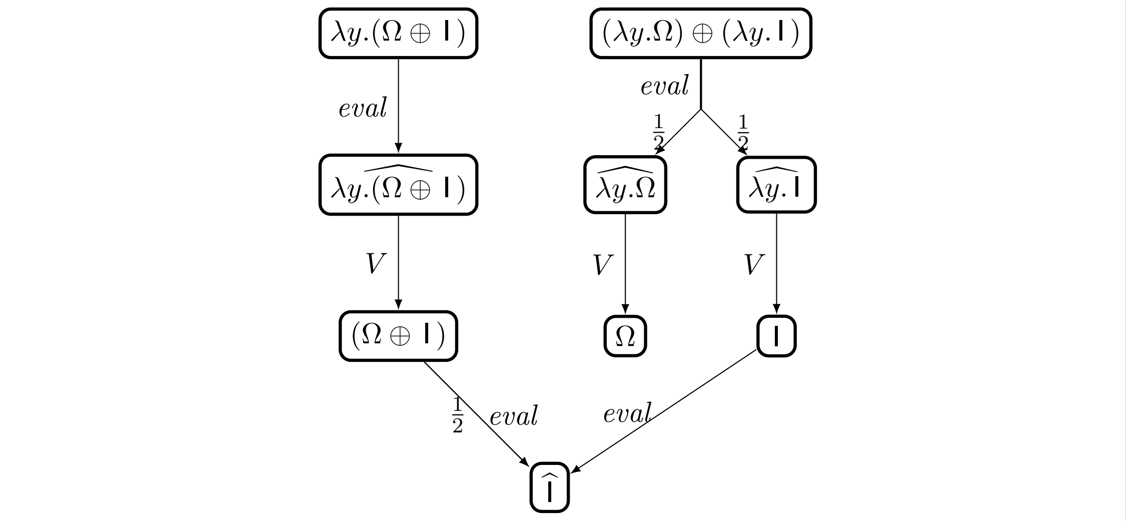
$\lambda y.(\Omega \oplus \mathsf{I})$
$(\lambda y.\Omega) \oplus (\lambda y.\mathsf{I})$
$eval$
$eval$
$\frac{1}{2}$
$\frac{1}{2}$
$\widehat{\lambda y.(\Omega \oplus \mathsf{I})}$
$\widehat{\lambda y.\Omega}$
$\widehat{\lambda y.\mathsf{I}}$
V
V
V
$(\Omega \oplus \mathsf{I})$
Ω
I
$\frac{1}{2}$
$eval$
$eval$
$\widehat{\mathsf{I}}$

Figure 7.1: RPLTS for M, N.

comparison of the weights of the $\mathcal{R}$ images of sets of states, as in [BMOW05; DLSA14; CD14].

Definition 7.3. Let $\mathcal{L} = (S, \mathcal{A}, \longrightarrow)$ be an RPLTS. A probabilistic simulation is a binary relation $\mathcal{R}$ on S such that if $(s_1, s_2) \in \mathcal{R}$ then for all $a \in \mathcal{A}$ it holds that $s_1 \xrightarrow{a} \Delta_1$ implies $s_2 \xrightarrow{a} \Delta_2$ with $\Delta_1(S') \leq \Delta_2(\mathcal{R}(S'))$ for all $S' \subseteq S$.

Requiring a simulation to be a preorder is not necessary, since the largest probabilistic simulation according to Definition 7.3 is a preorder, and it coincides with the union of all simulations. We let $\lesssim_{\mathrm{PS}}$ denote the simulation preorder based on this definition; probabilistic bisimilarity coincides with $\lesssim_{\mathrm{PS}} \cap \lesssim_{\mathrm{PS}}^{-1}$ [DLSA14]. The definition of simulation implies that whenever M is simulated by N we have that $\texttt{weight}(\llbracket M \rrbracket) \leq \texttt{weight}(\llbracket N \rrbracket)$. Analogously, if M is bisimilar to N, then $\texttt{weight}(\llbracket M \rrbracket) = \texttt{weight}(\llbracket N \rrbracket)$.

An applicative simulation $\mathcal{R}$ on terms of $\Lambda_{\oplus}$ can be easily seen as a simulation relation $\mathcal{R}'$ on states of $\mathcal{L}_{\oplus}$, obtained by adding to relation $\mathcal{R}$ the pairs $\{(\hat{V}, \hat{W}) \mid V \,\mathcal{R}\, W\}$. Analogously, a simulation relation on $\mathcal{L}_{\oplus}$ corresponds to an applicative simulation for $\Lambda_{\oplus}$.

Hence, we derive that on terms of $\Lambda_{\oplus}$, applicative similarity $\lesssim$ and bisimilarity $\approx$ coincide with $\lesssim_{\mathrm{PS}}$ and $\sim_{\mathrm{PB}}$ defined on the RPLTS $\mathcal{L}_{\oplus}$.

In what follows, we will often use the characterizations of simulation and bisimulation for the RPLTS $\mathcal{L}_{\oplus}$. Moreover, $\lesssim$ coincides with the simulation preorder defined in [CD14], which requires simulations to be preorders themselves. Consider now the terms M and N defined in Example 7.6 and represented in Figure 7.1 as states in $\mathcal{L}_{\oplus}$. Term M is not simulated by N: if a simulation $\mathcal{R}$ relates them, then it must also relate term $(\Omega \oplus \mathsf{I})$ to both term Ω and term I. However, $(\Omega \oplus \mathsf{I})$ can perform *eval* and reach I with probability one half, while Ω has zero probability of becoming a value, which means that $\mathcal{R}$ cannot be a simulation relation. In the other direction, we have that N cannot be simulated by M either. If $\mathcal{R}$ is simulation such that $N \,\mathcal{R}\, M$ then it must relate term I to term $(\Omega \oplus \mathsf{I})$, but the former has probability one of convergence and the latter has probability one half of convergence.

7.2 A probabilistic λ-calculus with parallel disjunction

In this section, we present the syntax and operational semantics of $\Lambda_{\oplus or}$, a λ-calculus endowed with probabilistic choice and parallel disjunction (or "parallel or") operators.

Definition 7.4. The terms of $\Lambda_{\oplus or}$ are expressions generated by the following grammar:

$$M, N, L = x \;\big|\; \lambda x.M \;\big|\; M \oplus N \;\big|\; M\,N \;\big|\; [M \parallel N] \rightarrowtail L$$

where $x \in \mathit{Var}$.

We let $FV(M)$ denote the set of free variables of the term M. A term M is closed if $FV(M) = \emptyset$. Given a set $\overline{x}$ of variables, $\Lambda_{\oplus or}(\overline{x})$ is the set of terms M such that $FV(M) \subseteq \overline{x}$. We write $\Lambda_{\oplus or}$ for $\Lambda_{\oplus or}(\emptyset)$.

The constructs of the λ-calculus have their usual meanings, and $M \oplus N$ is the binary, fair, probabilistic choice operator. The construct $[M \parallel N] \rightarrowtail L$ corresponds to the so-called parallel disjunction operator: if the evaluation of M or N terminates, then the behavior of $[M \parallel N] \rightarrowtail L$ is the same as the behavior of L, otherwise this term does not terminate. Since we are in a probabilistic calculus, this means that $[M \parallel N] \rightarrowtail L$ converges to L with a probability that is equal to the probability that either M or N converge. (This formulation of parallel disjunction is equivalent to the binary one, without the third term.)

The evaluation relation is the extension to $\Lambda_{\oplus or}$ of the evaluation relation presented in Figure 6.2 for the untyped probabilistic λ-calculus. Since the calculus has a call-by-value evaluation strategy, function arguments are evaluated before being passed to functions. Hence, the operational semantics is given by adding the rule in Figure 7.2 to the rules for the probabilistic call-by-value λ-calculus.

$$\textsc{Or}\ \frac{M \Downarrow \Delta \qquad N \Downarrow \Theta \qquad L \Downarrow \Xi}{[M \parallel N] \rightarrowtail L \Downarrow (\texttt{weight}(\Delta) + \texttt{weight}(\Theta) - (\texttt{weight}(\Delta) \cdot \texttt{weight}(\Theta))) \cdot \Xi}$$

Figure 7.2: Big-step semantics for parallel disjunction

Lemma 7.5. *For every term M, if $M \Downarrow \Delta$, and $M \Downarrow \Theta$, then there exists a distribution Ξ such that $M \Downarrow \Xi$ with $\Delta \leq \Xi$, and $\Theta \leq \Xi$.*

Proof. The proof is by induction on the structure of derivations for $M \Downarrow \Delta$. We only consider two cases, since the others are the same as in [DZ12]:

- If the derivation for $M \Downarrow \Delta$ is $\textsc{Empty}\ \frac{}{M \Downarrow \emptyset}$, then it is enough to take $\Xi = \Theta$, and since $\emptyset \leq \Theta$ and $\Theta \leq \Theta$ the result holds.
- If the derivation for $M \Downarrow \Delta$ is of the form:

$$\textsc{Or}\ \frac{P \Downarrow \Delta_1 \qquad N \Downarrow \Delta_2 \qquad L \Downarrow \Delta_3}{M = [P \parallel N] \rightarrowtail L \Downarrow \Delta = (\texttt{weight}(\Delta_1) + \texttt{weight}(\Delta_2) - (\texttt{weight}(\Delta_1) \cdot \texttt{weight}(\Delta_2))) \cdot \Delta_3}$$

Since $M = [P \parallel N] \rightarrowtail L$, there are only two possible structures for the derivation of $M \Downarrow \Theta$: either $\Theta = \emptyset$ and the result holds by $\Xi = \Delta$, or the structure of $M \Downarrow \Theta$ is the following:

$$\textsc{Or}\ \frac{P \Downarrow \Theta_1 \qquad N \Downarrow \Theta_2 \qquad L \Downarrow \Theta_3}{M = [P \parallel N] \rightarrowtail L \Downarrow \Theta = (\texttt{weight}(\Theta_1) + \texttt{weight}(\Theta_2) - (\texttt{weight}(\Theta_1) \cdot \texttt{weight}(\Theta_2))) \cdot \Theta_3}$$

By applying the induction hypothesis, we obtain that there exist Ξ_1, Ξ_2, Ξ_3 value distributions such that $P \Downarrow \Xi_1$, $N \Downarrow \Xi_2$, $L \Downarrow \Xi_3$, and, moreover, $\Delta_1, \Theta_1 \leq \Xi_1$, $\Delta_2, \Theta_2 \leq \Xi_2$, and $\Delta_3, \Theta_3 \leq \Xi_3$. We define $\Xi = (\texttt{weight}(\Xi_1) + \texttt{weight}(\Xi_2) - (\texttt{weight}(\Xi_1) \cdot \texttt{weight}(\Xi_2))) \cdot \Xi_3$, and we have that $M \Downarrow \Xi$. We must show that $\Delta \leq \Xi$ and $\Theta \leq \Xi$. Let $f : [0,1] \times [0,1] \to [0,1]$ be the function defined by $f(x,y) = x + y - x \cdot y$. The result follows from the fact that f is an increasing function, which holds since its two partial derivatives are positive.

□

Since distributions form an ω-complete partial order, and for every M the set of those distributions Δ such that $M \Downarrow \Delta$ is a countable directed set (by Lemma 7.5), the infinitary big-step semantics $[\![M]\!] = \sup_{M \Downarrow \Delta} \Delta$ is well-defined, and associates a unique value distribution to every term.

The definitions of probabilistic applicative bisimulation and simulation can be directly applied to $\Lambda_{\oplus or}$ and its operational semantics, and the RPLTS $\mathcal{L}_{\oplus or}$ for $\Lambda_{\oplus or}$ is defined as $\mathcal{L}_{\oplus}$. The contextual equivalence and preorder are as in Definition 6.12, using the contexts of $\Lambda_{\oplus or}$, which are defined by the following grammar:

$$\begin{aligned} C ::= x \;\big|\; [\cdot] \;\big|\; \lambda x.C \;\big|\; CM \;\big|\; MC \;\big|\; C \oplus M \;\big|\; M \oplus C \\ \big|\; [C \parallel M] \rightarrowtail N \;\big|\; [M \parallel C] \rightarrowtail N \;\big|\; [M \parallel N] \rightarrowtail C\,. \end{aligned}$$

Example 7.6. To see how things differ when we consider the contextual preorder in $\Lambda_{\oplus}$ and in $\Lambda_{\oplus or}$, consider the following terms of $\Lambda_{\oplus}$:

$$M = \lambda y.(\Omega \oplus \mathsf{I}) \qquad N = (\lambda y.\Omega) \oplus (\lambda y.\mathsf{I}).$$

where Ω and I are defined as in Example 6.10. We let $\leq_{\oplus}$ and $=_{\oplus}$ respectively denote the contextual preorder and equivalence for the language $\Lambda_{\oplus}$, i.e., the relations restricted to terms and contexts without the parallel disjunction construct. In [CD14] it is proved that $M \leq_{\oplus} N$. The converse does not hold, since if we take the $\Lambda_{\oplus}$ context

$$C = (\lambda x.(x\mathsf{I})(x\mathsf{I}))[\cdot]$$

we have that in $C[M]$ the term $\lambda y.(\Omega \oplus \mathsf{I})$ is copied with probability one, while in $C[N]$ both term $\lambda y.\Omega$ and term $\lambda y.\mathsf{I}$ are copied with probability one half. Hence, $C[M]$ converges with probability one quarter (i.e., the probability that $\Omega \oplus \mathsf{I}$ converges two times in a row) while $C[N]$ has probability one half of diverging (i.e., one half times the probability that Ω diverges two times in a row) and one half of converging (i.e., one half times the probability that I converges two times in a row). In $\Lambda_{\oplus or}$ we still have that $N \not\leq_{\mathtt{ctx}} M$, since the contexts of $\Lambda_{\oplus}$ are contexts of $\Lambda_{\oplus or}$ as well, but we also have that $M \not\leq_{\mathtt{ctx}} N$. Consider the context

$$C = (\lambda x.\,[(x\mathsf{I}) \parallel (x\mathsf{I})] \rightarrowtail \mathsf{I}\,)[\cdot]$$

If we put term M in context C then $\lambda y.(\Omega \oplus \mathsf{I})$ is copied, and $\lambda y.(\Omega \oplus \mathsf{I})$ has probability one half of converging when applied to I. Hence, by summing the probabilities of convergence of the two copies of $(\lambda y.(\Omega \oplus \mathsf{I}))\,\mathsf{I}$ and subtracting the probability that they both converge, we obtain that $[\![C[M]]\!] = \frac{3}{4} \cdot \texttt{dirac}(\mathsf{I})$. Term $C[N]$ only converges with probability one half, since with one half probability we have the parallel disjunction of two terms that never converge and with one half probability we have the parallel disjunction of two terms that always converge. Hence, both in $\Lambda_{\oplus}$ and in $\Lambda_{\oplus or}$ terms M, N are not contextually equivalent, but it is only in $\Lambda_{\oplus or}$ that neither M is below N nor N is below M in the contextual preorder. We will see in the following section that this corresponds to what happens when we consider the simulation preorder.

7.3 The simulation preorder is a precongruence

The extension $\precsim_\circ$ of the applicative simulation preorder to open terms is defined by considering all closing substitutions, i.e., for all $M, N \in \Lambda_{\oplus or}(x_1, \ldots, x_n)$, we have $M \precsim_\circ N$ if

$$M\{V_1, \ldots, V_n/x_1, \ldots, x_n\} \precsim_\circ N\{V_1, \ldots, V_n/x_1, \ldots, x_n\}, \text{ for all } V_1, \ldots, V_n \in \mathcal{V}_{\oplus or}.$$

Here we show that $\precsim_\circ$ is a precongruence, i.e., closed with respect to the operators of $\Lambda_{\oplus or}$.

It is here convenient to work with generalizations of relations called $\Lambda_{\oplus or}$-relations, i.e. sets of triples in the form $(\overline{x}, M, N)$, where $M, N \in \Lambda_{\oplus or}(\overline{x})$. Given a relation $\mathcal{R}$ on open terms, if $M \,\mathcal{R}\, N$ and $M, N \in \Lambda_{\oplus or}(\overline{x})$ then the triple $(\overline{x}, M, N)$ is in the corresponding $\Lambda_{\oplus or}$-relation. We denote this by $\overline{x} \vdash M \,\mathcal{R}\, N$. We extend the usual notions of symmetry, reflexivity and transitivity to $\Lambda_{\oplus or}$-relations as expected.

Definition 7.7. A $\Lambda_{\oplus or}$-relation $\mathcal{R}$ is compatible if and only if the following conditions hold:
(Com1) $\forall \overline{x}, \forall x \in \overline{x}, \overline{x} \vdash x \,\mathcal{R}\, x$;
(Com2) $\forall \overline{x}, \forall x \notin \overline{x}, \forall M, N, \overline{x} \cup \{x\} \vdash M \,\mathcal{R}\, N \implies \overline{x} \vdash \lambda x.M \,\mathcal{R}\, \lambda x.N$;
(Com3) $\forall \overline{x}, \forall M, N, P, Q, \overline{x} \vdash M \,\mathcal{R}\, N \wedge \overline{x} \vdash P \,\mathcal{R}\, Q \implies \overline{x} \vdash MP \,\mathcal{R}\, NQ$;
(Com4) $\forall \overline{x}, \forall M, N, P, Q, \overline{x} \vdash M \,\mathcal{R}\, N \wedge \overline{x} \vdash P \,\mathcal{R}\, Q \implies \overline{x} \vdash M \oplus P \,\mathcal{R}\, N \oplus Q$;
(Com5) $\forall \overline{x}, \forall M, N, P, Q, T, \overline{x} \vdash M \,\mathcal{R}\, N \wedge \overline{x} \vdash P \,\mathcal{R}\, Q \implies$
$\overline{x} \vdash [M \parallel P] \rightarrowtail T \;\mathcal{R}\; [N \parallel Q] \rightarrowtail T$.

It follows from these properties that a compatible relation is reflexive, since this holds by *(Com1)* on variables, and it is preserved by the other operators by *(Com2)-(Com5)*:

Proposition 7.8. *If a relation is compatible, then it is reflexive.*

7.3.1 Howe's method

The main idea of Howe's method consists in defining an auxiliary relation $\precsim_\circ^H$ such that it is easy to see that it is compatible, and then prove that $\precsim_\circ = \precsim_\circ^H$.

Definition 7.9. Let $\mathcal{R}$ be a relation. We define inductively the relation $\mathcal{R}^H$ by the rules in Figure 7.3.

We are now going to show that if the relation $\mathcal{R}$ we start from satisfies minimal requirements, namely that it is reflexive and transitive, then $\mathcal{R}^H$ is guaranteed to be compatible and to contain $\mathcal{R}$. This is a direct consequence of the following results, whose proofs are standard inductions:

$$\frac{\overline{x}\cup\{x\}\vdash x\,\mathcal{R}\,M}{\overline{x}\cup\{x\}\vdash x\,\mathcal{R}^H\,M} \qquad \frac{\overline{x}\cup\{x\}\vdash M\,\mathcal{R}^H\,N \quad \overline{x}\vdash \lambda x.N\,\mathcal{R}\,L}{\overline{x}\vdash \lambda x.M\,\mathcal{R}^H\,L}$$

$$\frac{\overline{x}\vdash M\,\mathcal{R}^H\,N \quad \overline{x}\vdash L\,\mathcal{R}^H\,P \quad \overline{x}\vdash NP\,\mathcal{R}\,R}{\overline{x}\vdash ML\,\mathcal{R}^H\,R}$$

$$\frac{\overline{x}\vdash M\,\mathcal{R}^H\,N \quad \overline{x}\vdash L\,\mathcal{R}^H\,P \quad \overline{x}\vdash N\oplus P\,\mathcal{R}\,R}{\overline{x}\vdash M\oplus L\,\mathcal{R}^H\,R}$$

$$\frac{\overline{x}\vdash M\,\mathcal{R}^H\,N \quad \overline{x}\vdash L\,\mathcal{R}^H\,P \quad \overline{x}\vdash [N\parallel P]\rightarrowtail T\;\mathcal{R}\,R}{\overline{x}\vdash [M\parallel L]\rightarrowtail T\;\mathcal{R}^H\,R}$$

Figure 7.3: Howe's construction

- Let $\mathcal{R}$ be a reflexive relation. Then $\mathcal{R}^H$ is compatible.
- Let $\mathcal{R}$ be transitive. Then:

$$\big(\overline{x}\vdash M\;\mathcal{R}^H\;N\big)\wedge(\overline{x}\vdash N\;\mathcal{R}\;L)\Rightarrow\big(\overline{x}\vdash M\;\mathcal{R}^H\;L\big) \tag{7.1}$$

- If $\mathcal{R}$ is reflexive, then $\overline{x}\vdash M\;\mathcal{R}\;N$ implies $\overline{x}\vdash M\;\mathcal{R}^H\;N$.

We can now apply Howe's construction to $\precsim_\circ$, since it is clearly reflexive and transitive. The properties above then tell us that $\precsim_\circ^H$ is compatible and that $\precsim_\circ\subseteq\precsim_\circ^H$. What we are left with, then, is proving that $\precsim_\circ^H$ is also a simulation.[9]

Lemma 7.10. *$\precsim_\circ^H$ is value-substitutive: for all terms M, N and values V, W such that $x\vdash M\precsim_\circ^H N$ and $\emptyset\vdash V\precsim_\circ^H W$, it holds that $\emptyset\vdash M\{V/x\}\precsim_\circ^H N\{W/x\}$*

Proof. By induction on the derivation of $x\vdash M\precsim_\circ^H N$. □

We also need an auxiliary, technical, lemma about probability assignments, that we will use in the proof of the Key Lemma (7.13).

Definition 7.11. $\mathbb{P}=\big(\{p_i\}_{1\le i\le n},\ \{r_I\}_{I\subseteq\{1,\dots,n\}}\big)$ is said to be a *probability assignment* if for every $I\subseteq\{1,..,n\}$, it holds that $\sum_{i\in I}p_i\le\sum_{J\cap I\neq\emptyset}r_J$.

Lemma 7.12 (Disentangling Sets)**.** *Let $P=\big(\{p_i\}_{1\le i\le n},\{r_I\}_{I\subseteq\{1,\dots,n\}}\big)$ be a probability assignment. Then for every non-empty $I\subseteq\{1,\dots,n\}$, and for every $k\in I$, there is an $s_{k,I}\in[0,1]$ satisfying the following conditions:*

- *for every I, it holds that $\sum_{k\in I}s_{k,I}\le 1$;*
- *for every $k\in 1,\dots,n$, it holds that $p_k\le\sum_{\{I\mid k\in I\}}s_{k,I}\cdot r_I$.*

The proof is an application of the Max-Flow Min-Cut Theorem, see e.g., [DLSA14; CD14].

Given a set of set of open terms X, let $\lambda x.X=\{\lambda x.M\mid M\in X\}$.

[9] In the proof of congruence for the probabilistic call-by-value λ-calculus presented in [CD14], the transitive closure of $\precsim_\circ^H$ is considered, since the definition of simulation required the relation to be preorder, which implies that the transitivity of $\precsim_\circ^H$ is needed. Since we relaxed the definition of simulation, this is not anymore necessary.

Lemma 7.13 (Key Lemma). *For all terms M, N, if $\emptyset \vdash M \precsim^H_\circ N$, then for every $\lambda x.X \subseteq \mathcal{V}_{\oplus or}$ it holds that $[\![M]\!]\,(\lambda x.X) \leq [\![N]\!]\left(\precsim_\circ \left(\lambda x. \precsim^H_\circ (X)\right)\right)$.*

Proof. We show that the inequality holds for every approximation of the semantics of M, which implies the result since the semantics is the supremum of the approximations. In particular, we prove by induction on the structure of the derivation of $M \Downarrow \Delta$ that, for any M, N, if $M \Downarrow \Delta$ and $\emptyset \vdash M \precsim^H_\circ N$, then for every $\lambda x.X \subseteq \mathcal{V}_{\oplus or}$ it holds that $\Delta\,(\lambda x.X) \leq [\![N]\!]\left(\precsim_\circ \left(\lambda x. \precsim^H_\circ (X)\right)\right)$. We consider separately every possible rule which can be applied at the bottom of the derivation:

- If the rule used corresponds to the fact that, for every term, the empty distribution is an approximate semantics, the derivation is: $\dfrac{}{M \Downarrow \emptyset}$ then $\Delta = \emptyset$, and for all set of values $\lambda x.X$, $\Delta(\lambda x.X) = 0$, and it concludes the proof.

- If M is a value $V = \lambda x.L$ and the last rule of the derivation is $\dfrac{}{V \Downarrow \mathtt{dirac}(V)}$ then $\Delta = \mathtt{dirac}(V)$ is the Dirac distribution for V and, by the definition of Howe's lifting, $\left(\emptyset \vdash \lambda x.L \precsim^H_\circ N\right)$ was derived by the following rule:

$$\frac{x \vdash L \precsim^H_\circ P \qquad \emptyset \vdash \lambda x.P \precsim_\circ N}{\emptyset \vdash \lambda x.L \precsim^H_\circ N}$$

It follows from the definition of applicative simulation and from $(\emptyset \vdash \lambda x.P \precsim_\circ N)$ that $1 = [\![N]\!](\precsim_\circ \{\lambda x.P\})$. Let $\lambda x.X \subseteq \mathcal{V}_{\oplus or}$. If $\lambda x.L \notin \lambda x.X$ then $\Delta(\lambda x.X) = 0$ and the thesis holds. Otherwise, $\Delta(\lambda x.X) = \Delta(\lambda x.L) = 1 = [\![N]\!](\precsim_\circ \{\lambda x.P\})$. It follows from $L \precsim^H_\circ P$ and from $\lambda x.L \in \lambda x.X$ that $\lambda x.P \in \lambda x.(\precsim^H_\circ X)$; hence, $[\![N]\!](\precsim_\circ \{\lambda x.P\}) \leq [\![N]\!](\precsim_\circ \lambda x.(\precsim^H_\circ X))$.

- If the derivation of $M \Downarrow \Delta$ is of the following form:

$$\frac{M_1 \Downarrow \Phi \qquad M_2 \Downarrow \Xi \qquad \{P\{V/x\} \Downarrow \Theta_{P,V}\}_{\lambda x.P \in \mathrm{supp}(\Phi), V \in \mathrm{supp}(\Xi)}}{M_1 M_2 \Downarrow \sum_{V \in \mathrm{supp}(\Xi)} \Xi(V) \left(\sum_{\lambda x.P \in \mathrm{supp}(\Phi)} \Phi(\lambda x.P) \cdot \Theta_{P,V}\right)}$$

Then $M = M_1 M_2$ and we have that the last rule used in the derivation of $\emptyset \vdash M \precsim^H_\circ N$ is:

$$\frac{\emptyset \vdash M_1 \precsim^H_\circ M_1' \qquad \emptyset \vdash M_2 \precsim^H_\circ M_2' \qquad \emptyset \vdash M_1' M_2' \precsim_\circ N}{\emptyset \vdash M_1 M_2 \precsim^H_\circ N}$$

Let $\mathrm{supp}(\Phi) = \{\lambda x.P_1, \dots, \lambda x.P_n\}$ and $K_i = \precsim_\circ \{\lambda x.L \mid x \vdash P_i \precsim^H_\circ L\}$ and, symmetrically, $\mathrm{supp}(\Xi) = \{V_1, \dots, V_l\}$ and $X_k = \precsim_\circ \{\lambda x.L \mid V_k = \lambda x.M'$ and $x \vdash M' \precsim^H_\circ L\}$. Then by the inductive hypothesis on $M_1 \Downarrow \Phi$ and $M_2 \Downarrow \Xi$ we have that $\Phi\left(\bigcup_{i \in I} \{\lambda x.P_i\}\right) \leq [\![M_1']\!]\left(\bigcup_{i \in I} K_i\right)$ for every $I \subseteq \{1, .., n\}$ and $\Xi(\bigcup_{k \in I} \{V_k\}) \leq [\![M_2']\!]\left(\bigcup_{k \in I} X_k\right)$ for every $I \subseteq \{1, .., l\}$.
Lemma 7.12 allows us to derive that for all $U \in \bigcup_{1 \leq i \leq n} K_i$ there exist probability

values $r_1^U, \ldots, r_n^U$ and for all $W \in \bigcup_{1\leq k\leq l} X_k$ there exist probability values $s_1^W, .., s_l^W$ such that:

$$\begin{array}{lll} [\![M_1']\!](U) \geq \sum\limits_{1\leq i\leq n} r_i^U & [\![M_2']\!](W) \geq \sum\limits_{1\leq k\leq l} s_k^W & \forall U \in \bigcup\limits_{1\leq i\leq n} K_i, W \in \bigcup\limits_{1\leq k\leq l} X_k \\ \Phi(\lambda x.P_i) \leq \sum\limits_{U\in K_i} r_i^U & \Xi(V_k) \leq \sum\limits_{W\in X_k} s_k^W & \forall\, 1 \leq i \leq n, 1 \leq k \leq l \end{array}$$

Hence, for every value $Z \in \mathcal{V}_{\oplus or}$, we have that:

$$\begin{aligned} \Delta(Z) &= \sum_{1\leq k\leq l} \Xi(V_k) \cdot \sum_{1\leq i\leq n} \Phi(\lambda x.P_i) \cdot \Theta_{P_i,V_k}(Z) \\ &\leq \sum_{1\leq k\leq l} \sum_{W\in X_k} s_k^W \cdot \sum_{1\leq i\leq n} \sum_{U\in K_i} r_i^U \cdot \Theta_{P_i,V_k}(Z) \end{aligned}$$

If $U = \lambda x.U' \in K_i$ then there exists S such that:

$$(2) \quad \emptyset \vdash \lambda x.S \precsim_\circ U \quad (3) \quad x \vdash P_i \precsim_\circ^H S$$

By (2), $\emptyset \vdash S\{W/x\} \precsim_\circ U'\{W/x\}$. By (3) and by Lemma 7.10, for $W \in X_k$ we have that $\emptyset \vdash P_i\{V_k/x\} \precsim_\circ^H S\{W/x\}$. It follows from (7.1) that $\emptyset \vdash P_i\{V_k/x\} \precsim_\circ^H U'\{W/x\}$. Hence, by the induction hypothesis applied to $P_i\{V_k/x\}$ we derive that $\Theta_{P_i,V_k}(\lambda x.X) \leq [\![U'\{W/x\}]\!](\precsim_\circ \lambda x.(\precsim_\circ^H X))$. Therefore,

$$\begin{aligned} \Delta(\lambda x.X) &\leq \sum_{1\leq k\leq l} \sum_{W\in X_k} s_k^W \cdot \sum_{1\leq i\leq n} \sum_{U\in K_i} r_i^U \cdot \Theta_{P_i,V_k}(\lambda x.X) \\ &\leq \sum_{W\in \bigcup\limits_{1\leq k\leq l} X_k} \sum_{U\in \bigcup\limits_{1\leq i\leq n} K_i} \Big(\sum_{\{k|W\in X_k\}} s_k^W\Big) \cdot \Big(\sum_{\{i|U\in K_i\}} r_i^U\Big) [\![L_{U,W}]\!](\precsim_\circ \lambda x.(\precsim_\circ^H X)) \\ &\leq \sum_{W\in \bigcup\limits_{1\leq k\leq l} X_k} \sum_{U\in \bigcup\limits_{1\leq i\leq n} K_i} [\![M_2']\!](W) \cdot [\![M_1']\!](U) \cdot [\![L_{U,W}]\!](\precsim_\circ \lambda x.(\precsim_\circ^H X)) \\ &\leq [\![M_1' M_2']\!](\precsim_\circ \lambda x.(\precsim_\circ^H X)) \end{aligned}$$

where $L_{U,W} = U'\{W/x\}$ for any U such that $U = \lambda x.U'$.
A detailed proof for this case is presented in Section 7.5.

- If $M \Downarrow \Delta$ is derived by:

$$\frac{M_1 \Downarrow \Delta_1 \qquad M_2 \Downarrow \Delta_2}{M_1 \oplus M_2 \Downarrow \frac{1}{2}\Delta_1 + \frac{1}{2}\Delta_2}$$

then $\emptyset \vdash M \precsim_\circ^H N$ is derived by:

$$\frac{\emptyset \vdash M_1 \precsim_\circ^H N_1 \qquad \emptyset \vdash M_2 \precsim_\circ^H N_2 \qquad \emptyset \vdash N_1 \oplus N_2 \precsim_\circ N}{\emptyset \vdash M_1 \oplus M_2 \precsim_\circ^H N}$$

By the inductive hypothesis, for $i \in \{1,2\}$ we have that for any $\lambda x.X \subseteq \mathcal{V}_{\oplus or}$,

$$\Delta_i(\lambda x.X) \leq [\![N_i]\!](\precsim_{\circ}\lambda x.(\precsim_{\circ}^{H} X))$$

Hence, the result follows from:

$$\tfrac{1}{2}\cdot\Delta_1(\lambda x.X) + \tfrac{1}{2}\cdot\Delta_2(\lambda x.X) \leq \tfrac{1}{2}\cdot[\![N_1]\!](\precsim_{\circ}\lambda x.(\precsim_{\circ}^{H} X)) + \tfrac{1}{2}\cdot[\![N_2]\!](\precsim_{\circ}\lambda x.(\precsim_{\circ}^{H} X))$$

- If the last rule applied in the derivation of $M \Downarrow \Delta$ is of the following form:

$$\frac{M_1 \Downarrow \Delta_1 \qquad M_2 \Downarrow \Delta_2}{[M_1 \parallel M_2] \rightarrowtail T \Downarrow (\mathtt{weight}(\Delta_1) + \mathtt{weight}(\Delta_2) - \mathtt{weight}(\Delta_1)\cdot\mathtt{weight}(\Delta_2))\cdot\mathtt{dirac}(T)}$$

then $M = [M_1 \parallel M_2] \rightarrowtail T$ and $\emptyset \vdash M \precsim_{\circ}^{H} N$ is derived by:

$$\frac{\emptyset \vdash M_1 \precsim_{\circ}^{H} N_1 \qquad \emptyset \vdash M_2 \precsim_{\circ}^{H} N_2 \qquad \emptyset \vdash [N_1 \parallel N_2] \rightarrowtail T \precsim_{\circ} N}{\emptyset \vdash [M_1 \parallel M_2] \rightarrowtail T \precsim_{\circ}^{H} N}$$

By inductive hypothesis on $M_1 \Downarrow \Delta_1$ we have that for any $\lambda x.X \subseteq \mathcal{V}_{\oplus or}$, $\Delta_1(\lambda x.X) \leq [\![N_1]\!](\precsim_{\circ}\lambda x.(\precsim_{\circ}^{H} X))$. Hence, for $\lambda x.X = \mathrm{supp}(\Delta_1)$ we have that:

$$\mathtt{weight}(\Delta_1) = \Delta_1(\lambda x.X) \leq [\![N_1]\!](\precsim_{\circ}\lambda x.(\precsim_{\circ}^{H} X)) \leq [\![N_1]\!](\mathrm{supp}([\![N_1]\!])) = \mathtt{weight}([\![N_1]\!])$$

and, symmetrically, by the inductive hypothesis on $M_2 \Downarrow \Delta_2$ we have $\mathtt{weight}(\Delta_2) \leq \mathtt{weight}([\![N_2]\!])$. Therefore,

$$\begin{aligned} &\mathtt{weight}(\Delta)_1 + \mathtt{weight}(\Delta)_2 - \mathtt{weight}(\Delta)_1\cdot\mathtt{weight}(\Delta)_2 \\ \leq\ &\mathtt{weight}([\![N_1]\!]) + \mathtt{weight}([\![N_2]\!]) - \mathtt{weight}([\![N_1]\!])\cdot\mathtt{weight}([\![N_2]\!]) \end{aligned}$$

Let $\lambda x.X \subseteq \mathcal{V}_{\oplus or}$. If $T \notin \lambda x.X$ then $\Delta = 0$ and the result follows. Otherwise, it follows from $T = \lambda x.T' \in \precsim_{\circ}\lambda x.(\precsim_{\circ}^{H} \{T'\})$ (since both $\precsim_{\circ}$ and $\precsim_{\circ}^{H}$ are reflexive) that

$$\begin{aligned} \Delta(\lambda x.X) = \Delta(\lambda x.T') &= \mathtt{weight}(\Delta)_1 + \mathtt{weight}(\Delta)_2 - \mathtt{weight}(\Delta)_1\cdot\mathtt{weight}(\Delta)_2 \\ &\leq \mathtt{weight}([\![N_1]\!]) + \mathtt{weight}([\![N_2]\!]) - \mathtt{weight}([\![N_1]\!])\cdot\mathtt{weight}([\![N_2]\!]) \\ &= [\![N]\!](\lambda x.T') = [\![N]\!](\precsim_{\circ}\lambda x.(\precsim_{\circ}^{H} X)) \end{aligned}$$

□

A consequence of the Key Lemma, then, is that relation $\precsim_{\circ}^{H}$ on closed terms is an applicative simulation, thus included in the largest one, namely $\precsim$. Hence, if M, N are open terms and $x_1,\ldots,x_n \vdash M \precsim_{\circ}^{H} N$ then it follows from Lemma 7.10 that for all $V_1,\ldots,V_n,W_1,\ldots,W_n$ such that $\emptyset \vdash V_i \precsim_{\circ}^{H} W_i$ we have that

$$\emptyset \vdash M\{V_1,\ldots,V_n/x_1,\ldots,x_n\} \precsim_{\circ}^{H} N\{W_1,\ldots,W_n/x_1,\ldots,x_n\}$$

which implies (by the reflexivity of $\precsim_{\circ}^{H}$ and by $\precsim_{\circ}^{H}\subseteq \precsim_{\circ}$ on closed terms) that for all $V_1,\ldots,V_n$ we have that

$$\emptyset \vdash M\{V_1,\ldots,V_n/x_1,\ldots,x_n\}\precsim_{\circ}N\{V_1,\ldots,V_n/x_1,\ldots,x_n\}$$

i.e., $M\precsim_{\circ}N$. Since $\precsim_{\circ}$ is itself included in $\precsim_{\circ}^{H}$, we obtain that $\precsim_{\circ} =\precsim_{\circ}^{H}$. Hence, it follows from the transitivity of $\precsim_{\circ}$ and from the fact that $\precsim_{\circ}^{H}$ is compatible that:

Theorem 7.14 (Congruence)**.** $\precsim_\circ$ *is a precongruence .*

The congruence of $\precsim_\circ$ allows us to prove that it is a sound with respect to the contextual preorder.

Theorem 7.15 (Soundness)**.** *If* $M \precsim_\circ N$ *then* $M \leq N$*.*

Proof. Let $M \precsim_\circ N$. Using Theorem 7.14, it can be easily proved by induction on C that for any context C it holds that $C[M] \precsim_\circ C[N]$. If $C[M] \precsim_\circ C[N]$ then $\texttt{weight}([\![C[M]]\!]) \leq \texttt{weight}([\![C[M]]\!])$, which implies the result. □

7.4 Full abstraction

As we have seen in Section 2.3.3, bisimilarity on reactive probabilistic processes is characterized by the language of tests $\mathbf{T}$, defined by the following grammar:

$$\mathbf{t}, \mathbf{u} ::= \omega \;\big|\; a.\mathbf{t} \;\big|\; \langle \mathbf{t}, \mathbf{u} \rangle$$

where $a \in \mathcal{A}$ ranges over the actions of the considered probabilistic transition system. This characterization is used in [CD14] to show that applicative bisimilarity on terms is fully abstract with respect to contextual equivalence.

This full-abstraction result is based on the fact that, when we consider the particular probabilistic transition system for the probabilistic λ-calculus defined in Section 7.1, any of these tests can actually be encoded by a context. However, the characterization of the simulation preorder requires to add disjunctive tests.

Definition 7.16. Let $\mathcal{L} = (S, \mathcal{A}, \rightarrow)$ be a RPLTS. The *test language* $\mathbf{T}_\vee$ is given by the grammar $\mathbf{t}, \mathbf{u} ::= \omega \;\big|\; a.\mathbf{t} \;\big|\; \langle \mathbf{t}, \mathbf{u} \rangle \;\big|\; \mathbf{t} \vee \mathbf{u}$, where $a \in \mathcal{A}$.

The probability of success of a test is defined as in Section 2.3.3, for ω, the test for actions $a.t$ and the conjunctive test $\langle \mathbf{t}, \mathbf{u} \rangle$. The probability of success of the disjunctive test corresponds to the probability that at least one of the two tests is successful:

$$\Pr(\mathbf{t} \vee \mathbf{u}, s) = \Pr(\mathbf{t}, s) + \Pr(\mathbf{u}, s) - \Pr(\mathbf{t}, s) \cdot \Pr(\mathbf{u}, s)$$

The following proposition characterizes the simulation preorder on RPLTSs by means of sets of tests.

Proposition 7.17 ([BMOW05])**.** *Let* $\mathcal{L} = (S, \mathcal{A}, \rightarrow)$ *be an RPLTS and let* $s, s' \in \mathcal{S}$*. Then* $s \lesssim s'$ *if and only if for every* $\mathbf{t} \in \mathbf{T}_\vee$ *it holds that* $Pr(\mathbf{t}, s) \leq Pr(\mathbf{t}, s')$*.*

Example 7.18. Consider the two terms $M = \lambda x.(\mathsf{I} \oplus \Omega)$ and $N = (\lambda x.\mathsf{I}) \oplus (\lambda x.\Omega)$ from Example 7.6. We already know that, since they do not verify $M \lesssim N$, there exists a test $\mathbf{t} \in \mathbf{T}_\vee$ whose success probability when executed on M is strictly greater that its success probability when executed on N. We can actually explicitly give such a test: let $\mathbf{t} = \mathit{eval}.(\mathsf{I}.\mathit{eval}.\omega \vee \mathsf{I}.\mathit{eval}.\omega)$. Then it holds that:

$$\Pr(\mathbf{t}, \lambda x.(\mathsf{I} \oplus \Omega)) = \frac{3}{4}; \qquad \Pr(\mathbf{t}, (\lambda x.\mathsf{I}) \oplus (\lambda x.\Omega)) = \frac{1}{2}.$$

7.4.1 From tests to contexts

It is shown in [CD14] that the applicative simulation preorder is not fully abstract for $\mathsf{PCFL}_\oplus$ with respect to the contextual preorder: a direct consequence is that disjunctive tests cannot be simulated by contexts. In other words, it is not possible to write a program that has access to two sub-programs, and terminates with a probability equal to the probability that at least one of its sub-programs terminates. The proof of [CD14] is based on an encoding from $\mathbf{T}$ to the set of contexts. We are going to extend it into two encodings from $\mathbf{T}_\vee$ to the set of contexts of $\Lambda_{\oplus or}$: one encoding (denoted by Enc) expresses the action of tests on states of the form M, and the other one (denoted by $\widehat{\mathrm{Enc}}$) on states of the form $\hat{V}$. The intuitive idea behind Enc and $\widehat{\mathrm{Enc}}$ is the following: if we take a test $\mathbf{t}$, its success probability starting from the state M is the same as the convergence probability of the context $\mathrm{Enc}(\mathbf{t})$ filled by M, and similarly, its success probability starting from the state $\hat{V}$ is the same as the convergence probability of the context $\mathrm{Enc}(\mathbf{t})$ filled by V.

We let $\mathscr{C}$ denote the set of all contexts of $\Lambda_{\oplus or}$.

Definition 7.19. Let $\widehat{\mathrm{Enc}} : \mathbf{T}_\vee \to \mathscr{C}$ and $\mathrm{Enc} : \mathbf{T}_\vee \to \mathscr{C}$ be defined by:

$$\begin{aligned} \mathrm{Enc}(\omega) &= \lambda x.[\cdot]; & \widehat{\mathrm{Enc}}(\omega) &= \lambda x.[\cdot];\\ \mathrm{Enc}(V.\mathbf{t}) &= \Omega[\cdot]; & \widehat{\mathrm{Enc}}(V.\mathbf{t}) &= \mathrm{Enc}(\mathbf{t})[([\cdot]V)];\\ \mathrm{Enc}(\mathit{eval}.\mathbf{t}) &= (\lambda x.\widehat{\mathrm{Enc}}(\mathbf{t})[x])[\cdot]; & \widehat{\mathrm{Enc}}(\mathit{eval}.\mathbf{t}) &= \Omega[\cdot];\\ \mathrm{Enc}(\mathbf{t} \vee \mathbf{u}) &= g(\mathrm{Enc}(\mathbf{t}), \mathrm{Enc}(\mathbf{u})); & \widehat{\mathrm{Enc}}(\mathbf{t} \vee \mathbf{u}) &= g(\widehat{\mathrm{Enc}}(\mathbf{t}), \widehat{\mathrm{Enc}}(\mathbf{u}));\\ \mathrm{Enc}(\langle \mathbf{t}, \mathbf{u} \rangle) &= f(\mathrm{Enc}(\mathbf{t}), \mathrm{Enc}(\mathbf{u})); & \widehat{\mathrm{Enc}}(\langle \mathbf{t}, \mathbf{u} \rangle) &= f(\widehat{\mathrm{Enc}}(\mathbf{t}), \widehat{\mathrm{Enc}}(\mathbf{u})); \end{aligned}$$

where $f, g : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$ are defined by:

$$\begin{aligned} f(C, D) &= (\lambda x.(\lambda y, z.\mathsf{I})(C[x\mathsf{I}])(D[x\mathsf{I}]))(\lambda x.[\cdot]);\\ g(C, D) &= (\lambda x.(\,[C[x\mathsf{I}] \parallel D[x\mathsf{I}]] \rightarrowtail \mathsf{I}\,)(\lambda x.[\cdot]). \end{aligned}$$

The apparently complicated structure of f and g comes from the fact that we chose not to build contexts with several holes, to highlight how unary contexts can mimic polyadic contexts in the calculus. Intuitively, we could say that $g(C, D)$ would correspond to a multihole context $[C \parallel D] \rightarrowtail \mathsf{I}$. Moreover, the encoding of the fragment of $\mathbf{T}_\vee$ corresponding to $\mathbf{T}$ does not use parallel disjunction, i.e., the image of $\mathbf{T}$ by the encoding is a subset of the contexts of $\Lambda_\oplus$. We can now apply this encoding to the test we defined in Example 7.18.

Example 7.20. Recall the test $\mathbf{t} = \mathit{eval}.(\mathsf{I}.\mathit{eval}.\omega \vee \mathsf{I}.\mathit{eval}.\omega)$ defined in Example 7.18. We can apply the encoding to this particular test:

$$\mathrm{Enc}(\mathbf{t}) = (\lambda x.\,(\lambda z.\,[(\lambda y.(\lambda w.y))z\mathsf{I}\,\mathsf{I} \parallel (\lambda y.(\lambda w.y))z\mathsf{I}\,\mathsf{I}] \rightarrowtail \mathsf{I}\,)\,(\lambda y.x))\,[\cdot].$$

We can see that if we consider the terms $M = \lambda x.(\mathsf{I} \oplus \Omega)$ and $N = (\lambda x.\mathsf{I}) \oplus (\lambda x.\Omega)$ defined in Example 7.6, the probability of convergence of the context $\mathrm{Enc}(\mathbf{t})$ is the probability of success of the test $\mathbf{t}$ with respect to M and N:

$$\Pr(\mathbf{t}, M) = \mathtt{weight}(\llbracket \mathrm{Enc}(\mathbf{t})[M] \rrbracket) \qquad \Pr(\mathbf{t}, N) = \mathtt{weight}(\llbracket \mathrm{Enc}(\mathbf{t})[N] \rrbracket).$$

Theorem 7.21. *Let* $\mathbf{t}$ *be a test in* $\mathbf{T}_\vee$. *For every closed term* M *and every closed value* V *it holds that:*

$$Pr(\mathbf{t}, M) = \texttt{weight}([\![Enc(\mathbf{t})[M]]\!]) \qquad Pr(\mathbf{t}, \hat{V}) = \texttt{weight}([\![\widehat{Enc}(\mathbf{t})[V]]\!]).$$

Proof. We prove the thesis by induction on the structure of $\mathbf{t}$.

- If $\mathbf{t} = \omega$, then for every closed term M and every closed value V, $\mathrm{Pr}(\omega, M) = \mathrm{Pr}(\omega, \hat{V}) = 1$, and we have defined $\mathrm{Enc}(\omega) = \widehat{\mathrm{Enc}}(\omega) = \lambda x.[\cdot]$. Since $\mathrm{Enc}(\omega)[M]$ and $\widehat{\mathrm{Enc}}(\omega)[V]$ are values, the weight of their semantics is 1, and so the result holds.
- If $\mathbf{t} = \langle \mathbf{u}_1, \mathbf{u}_2 \rangle$, we can directly adapt the construction proposed in [CD14] to the untyped case. By the inductive hypothesis, for $1 \leq i \leq 2$ it holds that for every closed term M and every closed value V,

  $$\mathrm{Pr}(\mathbf{u}_i, M) = \texttt{weight}([\![\mathrm{Enc}(\mathbf{u}_i)[M]]\!]) \qquad \mathrm{Pr}(\mathbf{u}_i, \hat{V}) = \texttt{weight}([\![\widehat{\mathrm{Enc}}(\mathbf{u}_i)[V]]\!]).$$

  The overall effect of f is to copy the content of the hole into the holes of the two contexts C and D. For any closed term M, we can express the convergence probability of $f(C, D)[M]$ as a function of the convergence probability of $C[M]$ and $D[M]$:

  $$\begin{aligned}\texttt{weight}([\![f(C,D)[M]]\!]) &= (\texttt{weight}([\![C[(\lambda x.M)\,\mathsf{I}]]\!])) \cdot (\texttt{weight}([\![D[(\lambda x.M)\,\mathsf{I}]]\!]))\\ &= (\texttt{weight}([\![C[M]]\!])) \cdot (\texttt{weight}([\![D[M]]\!]))\end{aligned}$$

  Recall that we have defined:

  $$\begin{aligned}\mathrm{Enc}(\langle \mathbf{u}_1, \mathbf{u}_2 \rangle) &= f(\mathrm{Enc}(\mathbf{u}_1), \mathrm{Enc}(\mathbf{u}_2))\\ \widehat{\mathrm{Enc}}(\langle \mathbf{u}_1, \mathbf{u}_2 \rangle) &= f(\widehat{\mathrm{Enc}}(\mathbf{u}_1), \widehat{\mathrm{Enc}}(\mathbf{u}_2))\end{aligned}$$

  We have that, for any closed term M, and any closed value V:

  $$\begin{aligned}\texttt{weight}([\![\mathrm{Enc}(\langle \mathbf{u}_1, \mathbf{u}_2 \rangle)[M]]\!]) &= \mathrm{Pr}(\mathbf{u}_1, M) \cdot \mathrm{Pr}(\mathbf{u}_2, M) = \mathrm{Pr}(\langle \mathbf{u}_1, \mathbf{u}_2 \rangle, M)\\ \texttt{weight}([\![\widehat{\mathrm{Enc}}(\langle \mathbf{u}_1, \mathbf{u}_2 \rangle)[V]]\!]) &= \mathrm{Pr}(\mathbf{u}_1, \hat{V}) \cdot \mathrm{Pr}(\mathbf{u}_2, \hat{V}) = \mathrm{Pr}(\langle \mathbf{u}_1, \mathbf{u}_2 \rangle, \hat{V})\end{aligned}$$

- Now the case $\mathbf{t} = \mathbf{u}_1 \vee \mathbf{u}_2$. By the inductive hypothesis, for all $1 \leq i \leq 2$ it holds that for every closed term M and every closed value V,

  $$\mathrm{Pr}(\mathbf{u}_i, M) = \texttt{weight}([\![\mathrm{Enc}(\mathbf{u}_i)[M]]\!]) \qquad \mathrm{Pr}(\mathbf{u}_i, \hat{V}) = \texttt{weight}([\![\widehat{\mathrm{Enc}}(\mathbf{u}_i)[V]]\!]).$$

  The definition of g allows us to show:

  $$\begin{aligned}\texttt{weight}([\![g(C,D)[M]]\!]) &= \texttt{weight}([\![C[M]]\!]) + \texttt{weight}([\![D[M]]\!])\\ &\quad - \texttt{weight}([\![C[M]]\!]) \cdot \texttt{weight}([\![D[M]]\!])\end{aligned}$$

  and now it is straightforward to see that:

  $$\begin{aligned}\texttt{weight}([\![\mathrm{Enc}(\mathbf{u}_1 \vee \mathbf{u}_2)[M]]\!]) &= \mathrm{Pr}(\mathbf{u}_1 \vee \mathbf{u}_2, M);\\ \texttt{weight}([\![\widehat{\mathrm{Enc}}(\mathbf{u}_1 \vee \mathbf{u}_2)[V]]\!]) &= \mathrm{Pr}(\mathbf{u}_1 \vee \mathbf{u}_2, \hat{V}).\end{aligned}$$

- If $\mathbf{t} = a.\mathbf{u}$, there are two different kinds of actions:

- when $a = \mathit{eval}$, we first consider $\widehat{\mathrm{Enc}}(\mathbf{t})$: since the *eval* action is relevant only for states of $\mathcal{L}_{\oplus or}$ which are terms (and not distinguished values), we want $\widehat{\mathrm{Enc}}(\mathbf{t})[V]$ to always diverge. Since $\widehat{\mathrm{Enc}}(\mathbf{t}) = \Omega[\cdot]$ and since $[\![\Omega]\!] = \emptyset$, we have that for any closed value V, $[\![\widehat{\mathrm{Enc}}(\mathbf{t})[V]]\!] = \emptyset$.
  Now, we consider $\mathrm{Enc}(\mathbf{t})$. By the inductive hypothesis, we know that:
  $$\Pr(\mathbf{u}, \hat{V}) = \mathtt{weight}([\![\widehat{\mathrm{Enc}}(\mathbf{u})[V]]\!]).$$
  We have defined: $\mathrm{Enc}(a.\mathbf{u}) = \lambda x.(\widehat{\mathrm{Enc}}(\mathbf{u})[x])[\cdot]$. Let be M a closed term. Then it holds that:
  $$\begin{aligned}\mathtt{weight}([\![\mathrm{Enc}(a.\mathbf{u})[M]]\!]) &= \sum_V [\![M]\!](V) \cdot \mathtt{weight}([\![\widehat{\mathrm{Enc}}(\mathbf{u})[V]]\!])\\ &= \sum_V [\![M]\!](V) \cdot \Pr(\mathbf{u}, \hat{V})\\ &= \sum_{\hat{V}} [\![\hat{M}]\!](\hat{V}) \cdot \Pr(\mathbf{u}, \hat{V})\\ &= \Pr(\mathbf{u}, M)\end{aligned}$$
- When $a = V$, with $V \in \mathcal{V}_{\oplus or}$, we consider first $\mathrm{Enc}(V.\mathbf{u})$. It has been designed to be a context which diverges whatever its argument is, and so we indeed have: $\Pr(V.\mathbf{u}, M) = 0 = \mathtt{weight}([\![\mathrm{Enc}(V.\mathbf{u})[M]]\!])$. Then we consider $\widehat{\mathrm{Enc}}(\mathbf{t})$. Recall that we have defined: $\widehat{\mathrm{Enc}}(V.\mathbf{u}) = \mathrm{Enc}(\mathbf{u})[[\cdot]V]$. Let $W = \lambda x.M$ be a closed value:
  $$\begin{aligned}\mathtt{weight}([\![\widehat{\mathrm{Enc}}(V.\mathbf{u})[W]]\!]) &= \mathtt{weight}([\![\mathrm{Enc}(\mathbf{u})[WV]]\!])\\ &= \Pr(\mathbf{u}, WV)\\ &= \Pr(\mathbf{u}, M\{^{V}\!/\!_{x}\}) && \text{since } [\![WV]\!] = [\![M\{^{V}\!/\!_{x}\}]\!]\\ &= \Pr(V.\mathbf{u}, W).\end{aligned}$$

□

Theorem 7.22. *$\lesssim$ is fully abstract with respect to the contextual preorder.*

Proof. We already know that $\lesssim$ is sound, that is $\lesssim \subseteq \leq_{\mathtt{ctx}}$. Hence, what is left to show is that $\leq_{\mathtt{ctx}} \subseteq \lesssim$, which follows from Theorem 7.21. Let M and N be two closed terms such that $M \leq_{\mathtt{ctx}} N$. We want to show that $M \lesssim N$. By the testing characterization of simulation, it is sufficient to show that, for every test $\mathbf{t} \in \mathbf{T}_{\vee}$, $\Pr(\mathbf{t}, M) \leq \Pr(\mathbf{t}, N)$. Then the result is a consequence of Theorem 7.21, since every test $\mathbf{t}$ of $\mathbf{T}_{\vee}$ can be encoded by a context of $\Lambda_{\oplus or}$.

□

7.5 Proofs

Proof of Theorem 7.13 - application case

We show the detailed proof of the application case of the Key Lemma.
If the derivation of $M \Downarrow \Delta$ is of the following form:

$$\frac{M_1 \Downarrow \Phi \qquad M_2 \Downarrow \Xi \qquad \{P\{{}^{V}\!/x\} \Downarrow \Theta_{P,V}\}_{\substack{\lambda x.P \in \mathrm{supp}(\Phi),\\ V \in \mathrm{supp}(\Xi)}}}{M_1 M_2 \Downarrow \sum_{V \in \mathrm{supp}(\Xi)} \Xi(V) \left(\sum_{\lambda x.P \in \mathrm{supp}(\Phi)} \Phi(\lambda x.P) \cdot \Theta_{P,V}\right)}$$

then $M = M_1 M_2$ and we have that the last rule used in the derivation of $\emptyset \vdash M \precsim_\circ^H N$ is:

$$\frac{\emptyset \vdash M_1 \precsim_\circ^H M_1' \qquad \emptyset \vdash M_2 \precsim_\circ^H M_2' \qquad \emptyset \vdash M_1' M_2' \precsim_\circ N}{\emptyset \vdash M_1 M_2 \precsim_\circ^H N}$$

We are first going to apply the induction hypothesis to the derivation of $M_1 \Downarrow \Phi$. The support of the value distribution Φ is a finite set, say $\mathrm{supp}(\Phi) = \{\lambda x.P_1, \ldots, \lambda x.P_n\}$. For every $\lambda x.P_i$, we define the set $K_i = \precsim_\circ \{\lambda x.L \mid x \vdash P_i \precsim_\circ^H L\}$. Now we can apply the induction hypothesis to the derivation of $M_1 \Downarrow \Phi$. Since we know that $\emptyset \vdash M_1 \precsim_\circ^H M_1'$, we derive from the induction hypothesis that for all $I \subseteq \{1, .., n\}$,

$$\Phi\left(\bigcup_{i \in I} \{\lambda x.P_i\}\right) \leq [\![M_1']\!]\left(\bigcup_{i \in I} K_i\right) \tag{7.2}$$

Inequation (7.2) allows us to apply Lemma 7.12. For every $i \in \{1, \ldots, n\}$, let $p_i = \Phi(\lambda x.P_i)$ and for every $I \subseteq \{1, .., n\}$ define $r_I = \sum\limits_{U \text{ s.t. } \{i \mid U \in K_i\} = I} [\![M_1']\!](U)$. We can see that $((p_i)_{1 \leq i \leq n}, (r_I)_{I \subseteq \{1, \ldots, n\}})$ is a probabilistic assignment. So we can conclude, by applying Lemma 7.12, that for $I \subseteq \{1, \ldots, n\}$ and for $i \in I$, there are $s_{i,I}$ that satisfy the conditions in the Lemma.

For every $U \in \bigcup\limits_{1 \leq i \leq m} K_i$ and for every $i \in \{1, \ldots, n\}$, define $r_i^U = s_{i, \{j \mid U \in K_j\}} \cdot [\![M_1']\!](U)$ if $i \in \{j \mid U \in K_j\}$ and $r_i^U = 0$ otherwise. We have that:

$$\begin{aligned} [\![M_1']\!](U) &\geq \sum_{1 \leq i \leq n} r_i^U & \qquad & \forall U \in \bigcup_{1 \leq i \leq n} K_i \\ \Phi(V_i) &\leq \sum_{U \in K_i} r_i^U & & \forall\, 1 \leq i \leq n \end{aligned}$$

In the same way, we can apply the inductive hypothesis to M_2. Let $\mathrm{supp}(\Xi) = \{V_1, \ldots, V_l\}$ and let $X_i = \precsim_\circ \{\lambda x.L \mid V_i = \lambda x.M' \text{ and } x \vdash M' \precsim_\circ^H L\}$. We have by the induction hypothesis that for all $I \subseteq \{1, .., l\}$, $\Xi(\{V_k \mid k \in I\}) \leq [\![M_2']\!]\left(\bigcup_{k \in I} X_k\right)$. Hence, for all $W \in \bigcup_{1 \leq k \leq l} X_k$, there exist l real numbers $s_1^W, .., s_l^W$, such that:

$$\begin{aligned} [\![M_2']\!](W) &\geq \sum_{1 \leq k \leq l} s_k^W & \qquad & \forall W \in \bigcup_{1 \leq k \leq l} X_k \\ \Xi(V_k) &\leq \sum_{W \in X_k} s_k^W & & \forall\, 1 \leq k \leq l \end{aligned}$$

For every value $Z \in \mathcal{V}_{\oplus or}$, we have that:

$$\Delta(Z) = \sum_{1 \leq k \leq l} \Xi(V_k) \sum_{1 \leq i \leq n} \Phi(\lambda x.P_i) \cdot \Theta_{P_i, V_k}(Z)$$

$$\le \sum_{1\le k\le l}\left(\sum_{W\in X_k} s_k^W\right)\sum_{1\le i\le n}\left(\sum_{U\in K_i} r_i^U\right)\cdot \Theta_{P_i,V_k}(Z)$$
$$= \sum_{1\le k\le l}\left(\sum_{W\in X_k} s_k^W\left(\sum_{1\le i\le n}\left(\sum_{U\in K_i} r_i^U \cdot \Theta_{P_i,V_k}(Z)\right)\right)\right)$$

We prove that for any $\lambda x.X \subseteq \mathcal{V}_{\oplus or}$, if $U = \lambda x.U' \in K_i$, and $1 \le i \le n$, and $W \in X_k$, then $\Theta_{P_i,V_k}(\lambda x.X) \le [\![U'\{W/x\}]\!](\lambda x. \precsim_\circ^H X)$.

Let $U \in K_i$ and $W \in X_k$ for some i,k such that $1 \le i \le n$ and $1 \le k \le l$. Then there exists S such that

$$\emptyset \vdash \lambda x.S \precsim_\circ U \tag{7.3}$$
$$x \vdash P_i \precsim_\circ^H S \tag{7.4}$$

Moreover, since $W \in X_k$ we have that:

$$\emptyset \vdash V_k \precsim_\circ^H W \tag{7.5}$$

By (7.3), $\emptyset \vdash S\{W/x\} \precsim_\circ U'\{W/x\}$, and by (7.4), (7.5) and Lemma 7.10 we have that $\emptyset \vdash P_i\{V_k/x\} \precsim_\circ^H S\{W/x\}$. It follows by (7.1) that $\emptyset \vdash P_i\{V_k/x\} \precsim_\circ^H U'\{W/x\}$. Hence, by the induction hypothesis applied to $P_i\{V_k/x\}$ we have $\Theta_{P_i,V_k}(\lambda x.X) \le [\![U'\{W/x\}]\!](\precsim_\circ \lambda x.(\precsim_\circ^H X))$. Then we derive:

$$\begin{aligned}
&\Delta(\lambda x.X)\\
&\le \sum_{1\le k\le l}\left(\sum_{W\in X_k} s_k^W\left(\sum_{1\le i\le n}\left(\sum_{U\in K_i} r_i^U \cdot \Theta_{P_i,V_k}(\lambda x.X)\right)\right)\right)\\
&\le \sum_{1\le k\le l}\left(\sum_{W\in X_k} s_k^W\left(\sum_{1\le i\le n}\left(\sum_{U\in K_i} r_i^U \cdot [\![L_{U,W}]\!](\precsim_\circ \lambda x.(\precsim_\circ^H X))\right)\right)\right)\\
&\le \sum_{W\in\left(\bigcup_{1\le k\le l} X_k\right)}\ \sum_{U\in\left(\bigcup_{1\le i\le n} K_i\right)}\left(\sum_{k \text{ s.t. } W\in X_k} s_k^W\right)\cdot\left(\sum_{i \text{ s.t. } U\in K_i} r_i^U\right)[\![L_{U,W}]\!](\precsim_\circ \lambda x.(\precsim_\circ^H X))\\
&\le \sum_{W\in\left(\bigcup_{1\le k\le l} X_k\right)}\ \sum_{U\in\left(\bigcup_{1\le i\le n} K_i\right)}\left([\![M_2']\!](W)\right)\cdot\left([\![M_1']\!](U)\right)[\![L_{U,W}]\!](\precsim_\circ \lambda x.(\precsim_\circ^H X))\\
&\le [\![M_1'M_2']\!](\precsim_\circ \lambda x.(\precsim_\circ^H X))
\end{aligned}$$

where $L_{U,W} = U'\{W/x\}$ for any U such that $U = \lambda x.U'$.

Chapter 8

Probabilistic environmental bisimulation

Applicative simulations and bisimulations are known to have some significant limitations, as we have seen in Chapter 6. With probabilities, the drawbacks of applicative bisimilarity are magnified: full abstraction with respect to contextual equivalence may fail also in a pure λ-calculus, and Howe's technique has to be enriched with non-trivial 'disentangling' properties for sets of real numbers, these properties themselves proved by modeling the problem as a flow network and then applying the Max-flow Min-cut Theorem (see Section 7.3.1 in the previous chapter).

The price to pay to go beyond these limitations is moving to a more complex definition of bisimulation, based on a notion of environment. In this chapter, we define environmental bisimulations for probabilistic higher-order languages. As representative calculi we consider call-by-name and call-by-value λ-calculi, and a (call-by-value) λ-calculus extended with higher-order references.

In Section 8.1, we discuss the main features of our definitions of environmental bisimulations for probabilistic calculi. We then present in Section 8.2 environmental bisimulations for pure call-by-name, establish basic properties including full abstraction for bisimilarity and similarity, and develop various up-to techniques. Section 8.3 is devoted to the pure call-by-value λ-calculus, and in Section 8.4 we study the extension with imperative features. In each case we derive full abstraction results for probabilistic environmental similarity and bisimilarity with respect to the contextual preorder and contextual equivalence, respectively.

8.1 Main features

We discuss here the main differences of our proposals in comparison with ordinary (i.e., non probabilistic) environmental (bi)simulations.

Static and dynamic environments In ordinary environmental bisimulation the values produced during the bisimulation game are placed into the environment, so that the observer can later play them at will during the bisimulation game. This schema is irrespective of the evaluation strategy (call-by-name or call-by-value), and is *the* distinguishing feature of environmental bisimulations over the applicative ones. 'Playing a term' means copying it. However, in the λ-calculus the copying possibilities for call-by-name and call-by-value are quite different. In call-by-name, evaluation only occurs in functional position and therefore the term resulting from the evaluation may not be copied. In call-by-value, in contrast, a term may be evaluated also in argument position, and then given as input to a function; thus copying is possible also after evaluation. The different copying behavior is well visible, for instance, in linear logic interpretations of call-by-name and call-by-value [MOTW99].

Now, as we have seen in the previous chapter, the semantics of probabilistic languages is sensitive to the copying operation; for instance the probability of success of an experiment, if non-trivial, may be lowered by playing the experiment several times. This has a strong impact on behavioral equivalences for call-by-name and call-by-value in probabilistic λ-calculi. As an example,

$$A \overset{\mathtt{def}}{=} \lambda x.(x \oplus \Omega) \quad \text{and} \quad B \overset{\mathtt{def}}{=} (\lambda x.x) \oplus (\lambda x.\Omega) \tag{8.1}$$

are contextually equivalent in call-by-name: if evaluated alone they always terminate; if evaluated with an argument, they return the argument with the same probability. More generally, in call-by-name abstraction distributes over probabilistic choice. In contrast, distributivity fails in call-by-value, exploiting the possibility of copying evaluated terms; e.g., the probabilities of termination for A and B are different in the context $(\lambda x.x\,(x\,\lambda y.y))[\cdot]$ (see Example 7.6).

To be able to express such behavioral differences, in our environmental bisimulations the values produced during the bisimulation game are placed into the environment *only* in call-by-value. We call such a value environment a *dynamic environment* because it may grow during the bisimulation game. It is precisely the use of the dynamic environment that allows us to separate the two terms A and B above. In probabilistic call-by-name, dynamic environments would break full abstraction for contextual equivalence. The only environment for call-by-name is *static*. The static environment for two compared objects F, G is a pair of λ-terms M, N, which are, intuitively, the initial λ-terms from which, using evaluation and interaction according to the bisimulation game, the objects F, G have been derived. This (small) static environment is sufficient to ensure that the congruence proof of the bisimilarity remains in the style of ordinary environmental bisimulation (i.e., it does not require sophisticated techniques such as Howe's). In short, the static environment reflects the copying possibility for terms before evaluation, whereas the dynamic environment reflects the copying possibility for values resulting from evaluation.

Formal sums In our probabilistic relations the objects compared are not plain λ-terms but *formal sums*, that are the objects produced by the semantics of a term. These are, intuitively, syntactic representations of probability distributions. As a consequence, environments are not just tuples of values, but formal sums of tuples of values. To see why related objects must be formal sums, consider again the terms A and B in (8.1): our environmental bisimulation for call-by-name equates A and B by relating A and the formal sum resulting from the evaluation of B. None of the components of the formal sum, $\lambda x.x$ and $\lambda x.\Omega$, could separately be related with A. (A form of bisimulation on formal sums, namely a probabilistic version of *logical bisimulation*, is already defined in [DLSA14] for call-by-name; its drawbacks are discussed in Section 9.1.)

In pure call-by-value λ-calculus, full abstraction for contextual equivalence would also hold without formal sums (i.e., relating plain λ-terms), for the same reason why, in the same language, applicative bisimilarity on plain terms is fully abstract [CD14]. We do not pursue this simplification of environmental bisimulations because it would be unsound in extensions of the calculus. For instance, consider the following terms of a probabilistic λ-calculus with store (again, an instance of distributivity):

$$H \stackrel{\mathsf{def}}{=} (\boldsymbol{\nu}\, x :=0)(\lambda.(M \oplus N)) \qquad K \stackrel{\mathsf{def}}{=} (\boldsymbol{\nu}\, x :=0)(\lambda.M \oplus \lambda.N)$$

where, as in Section 6.1.2, $(\boldsymbol{\nu}\, x :=0)$ indicates the creation of a new reference, initialized with 0, $\lambda.L$ is a thunk (i.e., $\lambda z.L$ for z not free in L), and where, using $L_1 \,\underline{\mathtt{seq}}\, L_2$ for the sequential evaluation of L_1 followed by L_2,

$$\begin{aligned} M &\stackrel{\mathsf{def}}{=} \mathtt{if}\ !x = 0\ \mathtt{then}\ (x := 1 \,\underline{\mathtt{seq}}\, \mathtt{true})\ \mathtt{else}\ \Omega \\ N &\stackrel{\mathsf{def}}{=} \mathtt{if}\ !x = 0\ \mathtt{then}\ (x := 1 \,\underline{\mathtt{seq}}\, \mathtt{false})\ \mathtt{else}\ \Omega\ . \end{aligned}$$

The terms M and N only differ at their first evaluation, when the fresh location l (that was created with value 0 and substituted to x) is set to 1 and M produces `true` whereas N produces `false`; thereafter l is 1 and both terms diverge. As a consequence, H and K are contextually equivalent: at their first evaluation they always terminate, each returning `true` and `false` with the same probability, and at later evaluations they always diverge.

To place H and K in a bisimulation, H has to be related with the formal sum obtained from the evaluation of K; again, the single components alone would be distinguished. Once more, this is a copying issue, due to the possibility of copying terms but not stores.

Big-step reduction, term closure, and congruence proof To achieve full abstraction, in the probabilistic case the bisimulation clauses have to use a big-step, rather than a small-step, reduction relation. As we have seen in Section 6.2, the semantics of the probabilistic λ-calculus is given by taking the supremum of its finitary big-step approximants. Consider once again the terms defined in Example 6.11:

$$P \stackrel{\mathsf{def}}{=} RR \quad \text{and} \quad Q \stackrel{\mathsf{def}}{=} \lambda.\Omega\,, \quad \text{for} \quad R \stackrel{\mathsf{def}}{=} \lambda x.(xx) \oplus Q \tag{8.2}$$

The terms P and Q are contextually equivalent. However, only by exploring the whole computation tree produced by P does one find out that the infinite number of leaves in the tree makes a probability 1 of obtaining Q (i.e., a formal sum made of a finite subset of the leaves of the tree would not be equivalent to Q).

When the reduction relation is small-step, as in ordinary environmental bisimulations [SKS11], the related terms need not be values, because a normalizing term need not produce a value in a single step and bisimulations must be closed under the reduction adopted. In contrast, as our environmental bisimulations are big-steps, the bisimulation game may be confined to values.

A more significant consequence of the adoption of big-step reductions concerns the congruence proof. In environmental bisimulations the proof of congruence goes by induction over contexts, as in proofs for first-order languages. For this, the proofs in the literature rely on a 'small-step' reduction relation. This allows a tight control over the syntax of the contexts, which fails with big-step reductions because in a higher-order language contexts may arbitrarily grow during reduction. The induction over contexts in the proofs of congruence for ordinary environmental bisimulations is replaced, in probabilistic environmental bisimulation, by an induction on the number of small-step reductions with which a big-step approximant is derived (possibly coupled with an induction on the size of a context), combined with two levels of continuity arguments. One level stems from the least fixed point construction employed in the definition of the infinitary big-step semantics on terms. The second level stems from a characterization of bisimilarity as the kernel of the similarity preorder and, in turn, as the kernel of a finitary similarity in which (on the challenger side) the big-step reduction relation employed is finite. The proof of the characterization with the finitary similarity makes use of least fixed-points via a saturation construction on formal sums where, intuitively, a formal sum is better than another formal sum if the former one conveys more accurate probabilistic information than the latter one.

Up-to techniques Our proofs and examples rely on a few enhancements of the bisimulation proof method ('bisimulations up-to'), some of which are extensions of common (bi)simulation enhancements, others are specific to probabilistic calculi. An example of the latter is 'simulation up-to lifting', whereby it is sufficient, in the coinductive game, that two derivative formal sums are in the probabilistic lifting of the candidate relation, rather than in the candidate relation itself.

While the bisimulations act on formal sums and use infinitary big-step reductions to values, we also explore coinductive games played on plain λ-terms and on finitary multi-step reductions to terms (not necessarily values) as sound proof techniques. In particular, we combine these with up-to context, so to be able to compare terms in the middle of their evaluation when a common context can be isolated and removed.

8.2 Probabilistic call-by-name λ-calculus

The terms of the probabilistic λ-calculus, as we have seen in Chapter 6, are generated by the following grammar:

$$M, N ::= x \;\Big|\; \lambda x.M \;\Big|\; MN \;\Big|\; M \oplus N$$

In probabilistic languages, the semantics of a term is usually a *(sub)distribution*, that is, a function that specifies the probabilities of all possible outcomes for that term. We define here an alternative semantics based on *formal sums*, i.e., syntactic representations of distributions. Formal sums allow us a tighter control on the manipulations of the operational semantics, which is important in various places of our coinductive definitions and proofs. Formal sums have the form

$$\textstyle\sum_{i \in I} p_i; M_i$$

where $0 < p_i \leq 1$, for each i, $\sum_{i \in I} p_i \leq 1$, and I is a (possibly infinite) indexing set. In a summand $p_i; M_i$ of a formal sum, p_i is its *probability value* (or *weight*), and M_i is its *term*. The terms of different summands of a formal sum need not be different. We extend some definitions for probability distributions to formal sums. The weight $\texttt{weight}(\sum_{i \in I} p_i; M_i)$ of a formal sum is $\sum_{i \in I} p_i$. We let F, G range over formal sums, and we write the empty formal sum as $\emptyset$ (i.e., the formal sum with no summands). We write $F = G$ if F and G are syntactically equal modulo a permutation of the summands and modulo the presence of $\emptyset$ as summand. We use '+' for binary sums, in the usual infix form, and sometimes apply it also to formal sums, as in $F + G$. We write the empty formal sum as $\emptyset$. *Value* *formal sums*, ranged over by Y, Z are formal sums in which the term of each summand is a value.

There is an obvious mapping from formal sums to distributions, whereby a formal sum F yields the distribution in which the probability of a term M is the sum of the weights with which M appears in summands of F. The mapping is not injective: in general, infinitely many formal sums yield the same distribution (because of possible duplicates in the terms of the summands of a formal sum). For instance, $\frac{1}{2}; M + \frac{1}{4}; M$ and $\frac{3}{4}; M$ are two different formal sums, that correspond to the same distribution assigning probability $\frac{3}{4}$ to term M and probability 0 to any N such that $N \neq M$.

We sometimes decompose formal sums using a lifting construction. Given formal sums $F_i = \sum_{j \in J_i} p_{i,j}; M_{i,j}$, for $i \in I$, we define

$$\textstyle\sum_{i \in I} p_i \cdot F_i \overset{\mathtt{def}}{=} \sum_{i \in I, j \in J_i} p_i \cdot p_{i,j}; M_{i,j} \ ,$$

with $p_i \cdot \emptyset = \emptyset$. The semantics of a term M, written $[\![M]\!]$, is a value formal sum produced as the supremum of the value formal sums obtained by finite computations starting from M, using a preorder $\leq_{\mathtt{apx}}$ on formal sums in which $F_1 \leq_{\mathtt{apx}} F_2$ if F_1 is an approximant of F_2 (in other words F_2 conveys more information than F_1); formally, $F_2 = F_1 + G$ for some G. The semantics is obtained in various steps, whose rules are presented in Figure 8.1:

1. a single-step reduction relation $\longrightarrow$ from terms to formal sums (where the evaluation contexts are the usual ones for call-by-name, i.e., $C := [\cdot] \mid CM$) ;

2. a multi-step reduction relation $\Longrightarrow$ from terms and formal sums to formal sums, from which a relation $\Longmapsto$ to value formal sums is extracted by retaining only the summands whose term is a value via the function `val`:

$$\mathtt{val}(\textstyle\sum_i p_i; M_i) \overset{\mathtt{def}}{=} \sum_{\{i \mid M_i \text{is a value}\}} p_i; M_i \ ;$$

3. the semantics $[\![\ ]\!]$, mapping terms and formal sums to value formal sums via the supremum construction.

If $M \Longrightarrow \sum_{i \in I} p_i; M_i$ then I is finite, and each i represents a 'possible world' of the probabilistic run of M, with probability p_i and outcome M_i. The subset of possible worlds in which M_i is a value makes for an approximant of M, and from such approximants the semantics of M is obtained.

Since value formal sums form an ω-complete partial order with respect to the $\leq_{\mathtt{apx}}$ preorder, and for every M the set of those value formal sums Y such that $M \Longmapsto Y$ is a countable directed set, the semantics $[\![M]\!]$ of a term M exists and is unique.

Relations $\Longrightarrow$ and $\Longmapsto$ are finitary in the sense that a derivation proof where one of such relations appears in the conclusion only contains a finite number of 'small steps' (relation $\longrightarrow$). When reasoning by induction, sometimes we will need to make such number explicit, therefore writing $\Longrightarrow_n$ and $\Longmapsto_n$, respectively.

Rule MulT, in contrast with MulFS, does not need a finitary condition on the indexing set because a formal sum obtained in a small step from a term may have at most two summands.

Additional notation. We introduce here some additional notation, allowing us to easily manipulate terms and formal sums. This simplified notation is only used in the proofs of our results; in the rest of the paper, we use the extended notation defined above. For $Y = \sum_i p_i; \lambda x.M_i$ and $F = \sum_i p_i; M_i$, we define:

single-step reduction relation from terms to formal sums

$$\textsc{Beta}\ \frac{}{(\lambda x.M)N \longrightarrow 1; M\{{}^{N}\!/x\}} \qquad \textsc{Sum}\ \frac{}{M_1 \oplus M_2 \longrightarrow \frac{1}{2}; M_1 + \frac{1}{2}; M_2}$$

$$\textsc{Eval}\ \frac{M \longrightarrow \sum_i p_i; M_i \quad C \text{ is an evaluation context}}{C[M] \longrightarrow \sum_i p_i; C[M_i]}$$

$$\text{Evaluation Contexts} \qquad C := [\cdot] \mid CM$$

multi-step reduction relation from terms to formal sums

$$\textsc{Mul0}\ \frac{}{M \Longrightarrow 1; M} \qquad \textsc{MulT}\ \frac{M \longrightarrow \sum_i p_i; M_i \qquad M_i \Longrightarrow F_i}{M \Longrightarrow \sum_i p_i{\cdot}F_i}$$

multi-step reduction relation from formal sums to formal sums:

$$\textsc{MulFS}\ \frac{M_i \Longrightarrow F_i}{\sum_{i\in I} p_i; M_i + G \Longrightarrow \sum_{i\in I} p_i{\cdot}F_i + G}\ I \text{ finite}$$

multi-step reduction relation from terms and formal sums to value formal sums

$$\textsc{MulVT}\ \frac{M \Longrightarrow F \qquad \mathtt{val}(F) = Y}{M \Longmapsto Y} \qquad \textsc{MulVFS}\ \frac{F \Longrightarrow F' \qquad \mathtt{val}(F') = Y}{F \Longmapsto Y}$$

the semantic mapping, from terms and formal sums to value formal sums

$$[\![M]\!] \overset{\mathtt{def}}{=} \sup\{Y \mid M \Longmapsto Y\} \qquad [\![F]\!] \overset{\mathtt{def}}{=} \sup\{Y \mid F \Longmapsto Y\}$$

Figure 8.1: Operational semantics for call-by-name

- $\lambda x.M \bullet P = M\{{}^{P}\!/x\}$;
- $Y \bullet P \overset{\mathtt{def}}{=} \sum_i p_i; M_i\{{}^{P}\!/x\}$;
- $C[F] \overset{\mathtt{def}}{=} \sum_i p_i; C[M_i]$;
- $FP \overset{\mathtt{def}}{=} \sum_i p_i; M_i P$.

Remark 8.1. By default, the results and definitions of environmental bisimulations we will present are (implicitly) stated for closed terms. They can be generalized to open terms in a standard way for bisimulations in λ-calculi [SP07a; KW06b; SKS11], essentially deriving properties between open terms M and N from the corresponding properties between the closed terms $\lambda\widetilde{x}.M$ and $\lambda\widetilde{x}.N$, for $\{\widetilde{x}\} \supseteq FV(M) \cup FV(N)$. We will often omit the word "closed" when referring to closed terms and values.

8.2.1 Environmental bisimulation

In call-by-name, a *probabilistic environmental relation* is a set of elements each of which is of the form (M, N) or $((M, N), Y, Z)$, where M, N, Y, Z are all closed, M, N are $\Lambda_\oplus$-terms and Y, Z value formal sums. Intuitively, in the former elements M and N are terms that we wish to prove equal, and in the latter elements Y and Z are value formal sums obtained from M and N via evaluations and interactions with the environment. We use $\mathcal{R}, \mathcal{S}$ to range over probabilistic environmental relations. In a triple $((M, N), Y, Z)$ the pair component (M, N) is the *static environment*, and Y, Z are the *tested formal sums.* We write $\mathcal{R}_{(M,N)}$ for the relation $\{(Y, Z) \mid ((M, N), Y, Z) \in \mathcal{R}\}$; we accordingly use the infix notation $Y\ \mathcal{R}_{(M,N)}\ Z$, and similarly for $M\ \mathcal{R}\ N$. In the remainder of the chapter, when discussing probabilistic environmental relations, bisimulations, simulations, or similar, we abbreviate 'probabilistic environmental' as 'PE', or even omit it when non-ambiguous. Static environments (that is, pairs of $\Lambda_\oplus$-terms) are ranged over by $\mathcal{E}$. If $\mathcal{E} = (M, N)$ then its *context closure*, written $\mathcal{E}^\star$, is the set of all pairs of the form $(C[M], C[N])$. We use a similar notation for the context closure of relations on λ-terms.

Remark 8.2 (Static environment)**.** Our results would also hold admitting arbitrary sets of pairs of $\Lambda_\oplus$-terms as static environments, rather then single pairs. We have chosen single pairs so to bring up the minimal requirement on static environments for our proofs to hold (notably the congruence for bisimilarity).

Definition 8.3 (Environmental bisimulation, call-by-name)**.** A PE relation $\mathcal{R}$ is a *(PE) bisimulation* if

1. $M\ \mathcal{R}\ N$ implies $[\![M]\!]\ \mathcal{R}_{(M,N)}\ [\![N]\!]$;
2. $\sum_i p_i; \lambda x.M_i\ \mathcal{R}_\mathcal{E} \sum_j q_j; \lambda x.N_j$ implies:
   (a) $\sum_i p_i = \sum_j q_j$;
   (b) for all $(P, Q) \in \mathcal{E}^\star$, $\sum_i p_i \cdot [\![M_i\{P/x\}]\!]\ \mathcal{R}_\mathcal{E} \sum_j q_j \cdot [\![N_j\{Q/x\}]\!]$.

We write $\approx$ for *(PE) bisimilarity*, the union of all PE bisimulations.

While $\approx$ is a PE relation, we are ultimately interested in comparing λ-terms ($M \approx N$ if $M\, \mathcal{R}\, N$ for some bisimulation $\mathcal{R}$).

Remark 8.4. Using the additional notation defined in Section 8.2, we can write the bisimulation clauses for formal sums as follows:

(2) $Y\ \mathcal{R}_\mathcal{E}\ Z$ implies:
   (a) $\texttt{weight}(Y) = \texttt{weight}(Z)$;
   (b) for all $(P, Q) \in \mathcal{E}^\star$, $[\![Y \bullet P]\!]\ \mathcal{R}_\mathcal{E}\ [\![Z \bullet Q]\!]$.

Example 8.5. We have

$$M \overset{\texttt{def}}{=} (\lambda.\lambda.\Omega) \oplus (\lambda.\Omega) \approx \lambda.(\lambda.\Omega \oplus \Omega) \overset{\texttt{def}}{=} N\,.$$

This is proved noting that $[\![M]\!] = \frac{1}{2}; \lambda.\lambda.\Omega + \frac{1}{2}; \lambda.\Omega$ and $[\![N]\!] = 1; N$, using the bisimulation $\mathcal{R}$ in which $M\, \mathcal{R}\, N$, $[\![M]\!]\ \mathcal{R}_{(M,N)}\ [\![N]\!]$, $\frac{1}{2}; \lambda.\Omega\ \mathcal{R}_{(M,N)}\ \frac{1}{2}; \lambda.\Omega$, and $\emptyset\ \mathcal{R}_{(M,N)}\ \emptyset$. Terms M, N could not be equated by a bisimulation that acted only on terms (ignoring formal sums), as neither $\lambda.\lambda.\Omega$ nor $\lambda.\Omega$ can be equated to N.

Definition 8.6 (Simulation)**.** In Definition 8.3, and in the remainder of the chapter for other definitions of probabilistic bisimulation, the corresponding *simulation* is obtained by replacing the equality '=' on the weights with '$\leq$'; thus in Definition 8.3, clause (2a) becomes $\sum_i p_i \leq \sum_j q_j$.

The union of all simulations, *similarity*, is written $\lesssim$.

Theorem 8.7.

1. $\approx$ *and* $\lesssim$ *are the largest bisimulation and simulation, respectively.*
2. $\lesssim$ *is a preorder, and* $\approx$ *an equivalence.*
3. $\approx = \lesssim \cap \lesssim^{-1}$.

Proof. 1. If $M \approx N$ then there is a bisimulation $\mathcal{R}$ such that $M \mathrel{\mathcal{R}} N$. Therefore, $((M,N), [\![M]\!], [\![N]\!]) \in \mathcal{R} \subseteq \approx$. Analogously, if $(\mathcal{E}, Y, Z) \in \approx$ then $(\mathcal{E}, Y, Z) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$ and the formal sums have the same weight and, for all $P, Q \in \mathcal{E}^\star$, $(\mathcal{E}, [\![Y \bullet P]\!], [\![Z \bullet Q]\!]) \in \mathcal{R} \subseteq \approx$. The same holds for simulation.

2. Identity is a simulation, hence $\lesssim$ is reflexive. If $\mathcal{R}, \mathcal{S}$ are simulations, then their relational composition

$$\begin{aligned}\mathcal{R}\ \mathcal{S} = &\{(M,N) \mid \exists P \text{ such that } M \mathrel{\mathcal{R}} P \mathrel{\mathcal{S}} N\}\\ &\cup \{((M,N), Y, Z) \mid \exists Y', P \text{ such that } Y \mathrel{\mathcal{R}}_{(M,P)} Y' \mathrel{\mathcal{S}}_{(P,N)} Z\}\end{aligned}$$

is a simulation. If $M \mathrel{\mathcal{R}} P \mathrel{\mathcal{S}} N$ then $[\![M]\!] \mathrel{\mathcal{R}_{(M,P)}} [\![P]\!] \mathrel{\mathcal{S}_{(P,N)}} [\![N]\!]$. Hence, $[\![M]\!] (\mathcal{R}\ \mathcal{S})_{(M,N)} [\![N]\!]$. If $Y \mathrel{\mathcal{R}}_{(M,P)} Y' \mathrel{\mathcal{S}_{(P,N)}} Z$ then $\texttt{weight}(Y) \leq \texttt{weight}(Y') \leq \texttt{weight}(Z)$ and for every C, $[\![Y \bullet C[M]]\!] \mathrel{\mathcal{R}_{(M,P)}} [\![Y' \bullet C[P]]\!] \mathrel{\mathcal{S}_{(P,N)}} [\![Z \bullet C[N]]\!]$. Then $\lesssim$ is transitive and reflexive. Analogously, $\approx$ is reflexive and transitive, and for any bisimulation $\mathcal{R}$ it holds that

$$\mathcal{R}^{-1} = \{(M,N) \mid N \mathrel{\mathcal{R}} M\} \cup \{((M,N), Y, Z) \mid Z \mathrel{\mathcal{R}}_{(N,M)} Y\}$$

is a bisimulation as well

3. The result follows from (1) and from the fact that the calculus is deterministic:

   (a) if $\mathcal{R}$ is a bisimulation then both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations;

   (b) to prove that $\lesssim \cap \lesssim^{-1} \subseteq \approx$, we show that $\lesssim \cap \lesssim^{-1}$ is a bisimulation. Let $\mathcal{R}$ and $\mathcal{S}$ be two simulations. If $M \mathrel{\mathcal{R}} N$ and $N \mathrel{\mathcal{S}} M$ then $[\![M]\!] \mathrel{\mathcal{R}_{(M,N)}} [\![N]\!]$ and $[\![N]\!] \mathrel{\mathcal{S}_{(N,M)}} [\![M]\!]$, which implies that $((M,N), [\![M]\!], [\![N]\!]) \in \lesssim \cap \lesssim^{-1}$. If $((M,N), Y, Z) \in \mathcal{R}$ and $((N,M), Z, Y) \in \mathcal{S}$ then for all C we have $[\![Y \bullet C[M]]\!] \mathrel{\mathcal{R}_{(M,N)}} [\![Z \bullet C[N]]\!]$ and $[\![Z \bullet C[N]]\!] \mathrel{\mathcal{S}_{(N,M)}} [\![Y \bullet C[M]]\!]$. Therefore, $((M,N), [\![Y \bullet C[M]]\!], [\![Z \bullet C[N]]\!]) \in \lesssim \cap \lesssim^{-1}$. Finally the clause on the weights holds by $((M,N), Y, Z) \in \mathcal{R}$ and $((N,M), Z, Y) \in \mathcal{S}$, which imply $\texttt{weight}(Y) \leq \texttt{weight}(Z)$ and $\texttt{weight}(Z) \leq \texttt{weight}(Y)$, respectively.

□

The bisimilarity, or similarity, is directly defined using the semantics of terms, which is a least-fixed point on top of big-step approximants. When proving properties about bisimilarity and similarity, therefore, we need to reason about such approximants. For this, we introduce a finite-step simulation in which the challenge reductions of the simulation game employ the big-step approximants (the relation $\Longmapsto$ of Figure 8.1). We cannot have characterizations of bisimilarity in terms of a finite-step *bi*-similarity because in general the weights of the approximants of two bisimilar terms are different, as shown in Example 8.8. Hence, to reason about bisimilarity we go through its characterization via similarity (Theorem 8.7), and then the characterization of similarity via the finite-step similarity (Corollary 8.12).

Example 8.8. Let P and Q be the terms discussed in (8.2) in Section 8.1. A bisimulation relating P and Q is

$$\{(P, Q), ((P, Q), \textstyle\sum_{n\geq 1} \frac{1}{2^n}; Q, 1; Q), ((P, Q), \emptyset, \emptyset)\} \ .$$

We could not prove the equality using finite-step approximants for bisimulation, since those for P are of the form $\sum_{1\leq n\leq m} \frac{1}{2^n}; Q$, for some m, and thus have a smaller total weight than the formal sum $1; Q$ immediately produced by Q.

Definition 8.9. A PE relation $\mathcal{R}$ is a *finite-step simulation* if

1. $M \mathrel{\mathcal{R}} N$ and $M \Longmapsto Y$ imply $Y \mathrel{\mathcal{R}_{(M,N)}} [\![N]\!]$;
2. $\sum_i p_i; \lambda x.M_i \mathrel{\mathcal{R}_{\mathcal{E}}} \sum_j q_j; \lambda x.N_j$ implies:
   (a) $\sum_i p_i \leq \sum_j q_j$;
   (b) for all $(P, Q) \in \mathcal{E}^\star$, if $\sum_i p_i; M_i\{^P\!/\!_x\} \Longmapsto Y$ then $Y \mathrel{\mathcal{R}_{\mathcal{E}}} \sum_j q_j{\cdot}[\![N_j\{^Q\!/\!_x\}]\!]$.

We write $\lesssim_{\text{fin}}$ for *finite-step similarity.* In finite-step simulations, the challenges are expressed by finitary reductions. Moreover, any result about finite-step similarity $\lesssim_{\text{fin}}$ on $\Lambda_\oplus$-terms can be established using a finite-step simulation with finite formal sums on the challenger side, though this constraint is not required in the definition.

Remark 8.10. Clause (2b) of Definition 8.9 cannot be written thus:
for all $(P, Q) \in \mathcal{E}^\star$, if $M_i\{^P\!/\!_x\} \Longmapsto Y_i$ for every i
then $\sum_i p_i{\cdot}Y_i \mathrel{\mathcal{R}_{\mathcal{E}}} \sum_j q_j{\cdot}[\![N_j\{^Q\!/\!_x\}]\!]$
because, as the index set I can be infinite, the challenge in the bisimulation game might not be finitary. By contrast, reduction $\Longmapsto$ on formal sums (from Figure 8.1) is finitary. This allows proofs by induction on the number of single-steps in a reduction.

We denote by $\texttt{Pairs}(\mathcal{R})$ the set of pairs of terms in a PE relation $\mathcal{R}$. We use two saturation constructions to turn a simulation into a finite-step simulation and conversely. Given a PE relation $\mathcal{R}$, its *saturation by approximants* is

$$\texttt{Pairs}(\mathcal{R}) \cup \{(\mathcal{E}, Y, Z) \mid \text{there is } Y' \text{ with } Y' \mathrel{\mathcal{R}_{\mathcal{E}}} Z \text{ and } Y \leq_{\texttt{apx}} Y' \}$$

and its *saturation by suprema* is $\bigcup_n \mathcal{R}^n$, where

$$\begin{aligned} \mathcal{R}^0 &\stackrel{\texttt{def}}{=} \mathcal{R} \\ \mathcal{R}^{n+1} &\stackrel{\texttt{def}}{=} \mathcal{R}^n \cup \{(\mathcal{E}, Y, Z) \mid \text{there are } \{Y_i\}_i \text{ with} \\ &\qquad\qquad Y_i \mathrel{\mathcal{R}^n_{\mathcal{E}}} Z,\ \ Y_i \leq_{\texttt{apx}} Y_{i+1},\ \text{ and } Y = \sup\{Y_i\}_i\}. \end{aligned}$$

Lemma 8.11.

1. *The saturation by approximants of a simulation is a finite-step simulation.*
2. *The saturation by suprema of a finite-step simulation is a simulation.*

Proof. The proof of (1) follows from the definition of $[\![M]\!]$ as the supremum of the set $\{Y \mid M \Longmapsto Y\}$. For (2), the crux is proving by induction on n that if $\sum_i p_i; \lambda x.M_i \ \mathcal{R}^n_{\mathcal{E}} \ \sum_j q_j; \lambda x.N_j$ then:

1. $\sum_i p_i \leq \sum_j q_j$;
2. $\sum_i p_i; M_i\{P/x\} \Longmapsto Y$ implies $Y \ \mathcal{R}^n_{\mathcal{E}} \ \sum_j q_j \cdot [\![N_j\{Q/x\}]\!]$, for all $(P, Q) \in \mathcal{E}^\star$.

The details of the proof can be found in Section 8.5. □

Corollary 8.12. *The similarity and finite-step similarity preorders, $\lesssim$ and $\lesssim_{\text{fin}}$, coincide.*

Proof. The result follows from Lemma 8.11 and from the fact that a simulation (respectively: a finite-step simulation) is included in its saturation by approximants (respectively: by suprema). □

The following example highlights the differences between simulations and finite-step simulations, by proving the equality in Example 8.8 using finite-step simulations.

Example 8.13. Terms P and Q of Example 8.8 can be proved equivalent by exhibiting the following finite-step simulations, where $Y_0 \overset{\text{def}}{=} \emptyset$ and $Y_m \overset{\text{def}}{=} \sum_{1 \leq n \leq m} \frac{1}{2^n}; Q$ for $m \geq 1$:
$\mathcal{R} \overset{\text{def}}{=} \{(P, Q), ((P, Q), \emptyset, \emptyset)\} \cup \{((P, Q), Y_m, 1; Q) \mid m \geq 0\}$
$\mathcal{S} \overset{\text{def}}{=} \{(Q, P), ((Q, P), \emptyset, \emptyset), ((Q, P), 1; Q, \sum_{n \geq 1} \frac{1}{2^n}; Q)\}$. □

To derive the substitutivity properties of the similarity, and hence of the bisimilarity, we also need an up-to technique for the finite-step similarity. Specifically, we need an up-to lifting technique whereby, in the simulation game, two derivative formal sums can be decomposed into 'smaller' formal sums and it is then sufficient that these are pairwise related. We write $\texttt{lift}(\mathcal{S})$ for the probabilistic lifting of a relation $\mathcal{S}$ on formal sums:[10]

$$\begin{aligned}\texttt{lift}(\mathcal{S}) \overset{\text{def}}{=} \{(F, G) \mid \ & \text{there are } I,\ p_i, F_i, G_i, \text{ for } i \in I, \text{ with} \\ & F_i \ \mathcal{S} \ G_i \text{ and } F = \textstyle\sum_i p_i \cdot F_i \text{ and } G = \textstyle\sum_i p_i \cdot G_i\}.\end{aligned}$$

Definition 8.14. A PE relation $\mathcal{R}$ is a *finite-step simulation up-to lifting* if

1. $M \, \mathcal{R} \, N$ and $M \Longmapsto Y$ imply $Y \ \texttt{lift}(\mathcal{R}_{(M,N)}) \ [\![N]\!]$;
2. $\sum_i p_i; \lambda x.M_i \ \mathcal{R}_{\mathcal{E}} \ \sum_j q_j; \lambda x.N_j$ implies:
   (a) $\sum_i p_i \leq \sum_j q_j$;
   (b) for all $(P, Q) \in \mathcal{E}^\star$, if $\sum_i p_i; M_i\{P/x\} \Longmapsto Y$ then $Y \ \texttt{lift}(\mathcal{R}_{\mathcal{E}}) \ \sum_j q_j \cdot [\![N_j\{Q/x\}]\!]$.

[10] In contrast with the lifting relation defined in Chapter 2 (definition 2.10), the lifting relation defined here takes a relation on formal sums and returns a relation on formal formal sums. The lifting relation in Chapter 2 takes a relation on states (corresponding to terms in the setting of the present chapter) and returns a relation on distributions on states (formal sums in the present chapter).

Lemma 8.15. *If $\mathcal{R}$ is a finite-step simulation up-to lifting then $\mathcal{R} \subseteq \lesssim_{\text{fin}}$.*

Proof. Let $\mathcal{R}$ be a finite-step simulation up-to lifting. Then $\mathcal{S}$ is finite-step simulation:

$$\mathcal{S} \overset{\text{def}}{=} \texttt{Pairs}(\mathcal{R}) \cup \{((M,N),Y,Z) \mid Y\,\texttt{lift}(\mathcal{R}_{(MN)})\,Z\}$$

If $M\ \mathcal{S}\ N$ then $M\,\mathcal{R}\,N$, which implies that if $M \Longmapsto Y$ then $Y\,\texttt{lift}(\mathcal{R}_{(M,N)})\,[\![N]\!]$. Hence, $Y\ \mathcal{S}_{(M,N)}\ [\![N]\!]$.
Let $Y\ \mathcal{S}_{(M,N)}\ Z$, i.e., $Y = \sum_i p_i{\cdot}Y_i$, $Z = \sum_i p_i{\cdot}Z_i$ and $Y_i\,\mathcal{R}_{(M,N)}Z_i$. For every i, $\texttt{weight}(Y_i) \le \texttt{weight}(Z_i)$, hence $\texttt{weight}(Y) \le \texttt{weight}(Z)$.
For every i, $Y_i \bullet C[M] \Longmapsto Y_i'$ implies $Y_i'\,\texttt{lift}(\mathcal{R}_{(M,N)})\,[\![Z_i \bullet C[N]]\!]$ by the definition of $\mathcal{R}$. Since $Y \bullet C[M] \Longmapsto Y'$ implies $Y' = \sum_i p_i{\cdot}Y_i'$, for some Y_i' such that $Y_i \bullet C[M] \Longmapsto Y_i'$, and $[\![Z \bullet C[N]]\!] = \sum_i p_i; [\![Z_i \bullet C[N]]\!]$, then $Y'\,\texttt{lift}(\texttt{lift}(\mathcal{R}_{(M,N)}))\,[\![Z \bullet C[N]]\!]$. The result follows from $\texttt{lift}(\texttt{lift}(\mathcal{R}_{(M,N)})) = \texttt{lift}(\mathcal{R}_{(M,N)})$. □

Example 8.16. Let $P \overset{\text{def}}{=} \lambda.\Omega$, $Q \overset{\text{def}}{=} \lambda.\lambda.\Omega$, and

$$M \overset{\text{def}}{=} (P \oplus P) \oplus (Q \oplus Q)\ , \quad N \overset{\text{def}}{=} (P \oplus Q) \oplus (P \oplus Q)\,.$$

The following finite-step simulation up-to lifting shows $M {\lesssim_{\text{fin}}} N$:

$$\begin{aligned}\mathcal{R} \overset{\text{def}}{=} \{&(M,N),((M,N),1;P,1;P),((M,N),1;Q,1;Q),\\ &((M,N),\emptyset,1;P),((M,N),\emptyset,1;Q)\}\ .\end{aligned}$$

The 'up-to lifting' technique allows us to have a relation with only empty or Dirac formal sums (i.e., a single summand with probability 1).

Remark 8.17. We have seen in Example 8.8 that the terms P and Q defined in (8.2), Section 8.1, cannot be proved equivalent using a bisimulation with small-step, finitary clauses. We could prove the terms equivalent by using a bisimulation with small-step clauses if we allowed the reached formal sums to be decomposed into equally weighted formal sums (formally: using the up-to-distribution-and-lifting technique discussed in Section 8.2.3). In this case, it would be sufficient to define a bisimulation relating P and Q, and we would only need to consider the formal sum $\frac{1}{2}; P + \frac{1}{2}; Q$ (reached by P in one step) and decompose $1; Q$ (the formal sum reached in zero steps by Q) as $\frac{1}{2}; Q + \frac{1}{2}; Q$.
However, such a bisimulation would not be complete, since the same reasoning would not apply to the terms M and N defined below, where $R_1 \overset{\text{def}}{=} \lambda x.(xx) \oplus \lambda x.Q$ and $R_2 \overset{\text{def}}{=} \lambda x.(xx) \oplus Q$

$$M \overset{\text{def}}{=} (R_1 R_1) \oplus \lambda.\lambda.Q \quad N \overset{\text{def}}{=} \lambda.(R_2 R_2) \oplus \lambda.Q$$

Terms M and N cannot be proved equivalent using a small-step bisimulation, since $1; N$ is only equivalent to the semantics of M, which is not reachable in a finite number of steps. Indeed, no decomposition of $1; N$ can be matched with a decomposition of any approximation of the semantics of M.

Lemma 8.18. *If $M {\lesssim_{\text{fin}}} N$ then $C[M] {\lesssim_{\text{fin}}} C[N]$, for any context C.*

Proof. Given a a finite-step simulation $\mathcal{R}$ saturated by approximants, we prove that the PE relation

$$\begin{aligned}&\{(C[M], C[N]) \mid M\,\mathcal{R}\,N\}\\ &\cup\{((C[M], C[N]), 1; \lambda x.C'[M], 1; \lambda x.C'[N]) \mid M\,\mathcal{R}\,N\}\\ &\cup\{((C[M], C[N]), Y, Z) \mid Y\,\mathcal{R}_{(M,N)}Z\}\\ &\cup\{((M, N), \emptyset, Z) \mid \text{ for some } M, N, Z\}\end{aligned}$$

is a finite-step simulation up-to lifting. The details of the proof can be found in Section 8.5. □

Hence, (pre)congruence results for bisimilarity and similarity follow from Lemma 8.18, Corollary 8.12 and the fact that $\approx = \lesssim \cap \lesssim^{-1}$ (Theorem 8.7).

Corollary 8.19. *On $\Lambda_\oplus$-terms, $\approx$ is a congruence, and $\lesssim$ a precongruence.*

8.2.2 Contextual equivalence

The contextual preorder and equivalence are defined as in Section 6.2.2, using the weight of the semantics of terms, which are now formal sums. We set $M \Downarrow \stackrel{\text{def}}{=} \texttt{weight}(\llbracket M \rrbracket)$ (the probability of termination).

Definition 8.20 (Contextual preorder and equivalence)**.** M and N are in the *contextual preorder*, written $M \leq_{\texttt{ctx}} N$, (resp. in *contextual equivalence*, written $M =_{\texttt{ctx}} N$), if $C[M] \Downarrow \leq C[N] \Downarrow$ (resp. $C[M] \Downarrow = C[N] \Downarrow$), for every context C.

Lemma 8.21 (Completeness)**.** *On $\Lambda_\oplus$-terms, $\leq_{\texttt{ctx}} \subseteq \lesssim$.*

Proof. We prove that the following is a simulation:

$$\mathcal{R} \stackrel{\text{def}}{=} (\leq_{\texttt{ctx}}) \cup \{((M, N), \llbracket C[M] \rrbracket, \llbracket C[N] \rrbracket) \mid M \leq_{\texttt{ctx}} N\}$$

We have $M\ \mathcal{R}\ N$ if and only if $M \leq_{\texttt{ctx}} N$, which by definition of $\mathcal{R}$ implies that $\llbracket M \rrbracket\ \mathcal{R}_{(M,N)}\ \llbracket N \rrbracket$.
If $Y\ \mathcal{R}_{(M,N)}\ Z$ then $Y = \llbracket C[M] \rrbracket$ and $Z = \llbracket C[N] \rrbracket$ for some context C. Hence, for any C' we have $\llbracket Y \bullet C'[M] \rrbracket = \llbracket \llbracket C[M] \rrbracket \bullet C'[M] \rrbracket = \llbracket C[M]C'[M] \rrbracket$ and $\llbracket Z \bullet C'[N] \rrbracket = \llbracket \llbracket C[N] \rrbracket \bullet C'[N] \rrbracket = \llbracket C[N]C'[N] \rrbracket$ and by the definition of $\mathcal{R}$ we have $\llbracket C[M]C'[M] \rrbracket\,\mathcal{R}_{(M,N)} \llbracket C[N]C'[N] \rrbracket$. □

Corollary 8.22 (Full abstraction)**.** *On $\Lambda_\oplus$-terms:*

1. *relations $\leq_{\texttt{ctx}}$ and $\lesssim$ coincide.*

2. *relations $=_{\texttt{ctx}}$ and $\approx$ coincide.*

Proof. Completeness of the simulation preorder holds by Lemma 8.21. The converse, i.e., soundness, follows from the fact that $\lesssim$ is a precongruence (Corollary 8.19) and $M \lesssim N$ implies $\texttt{weight}(\llbracket M \rrbracket) \leq \texttt{weight}(\llbracket N \rrbracket)$ (by clause (2a) of simulation). Hence, $M \lesssim N$ implies $C[M] \lesssim C[N]$ implies $\texttt{weight}(\llbracket C[M] \rrbracket) \leq \texttt{weight}(\llbracket C[N] \rrbracket)$.
Completeness and soundness for $\approx$ follow from (1) and the fact that $\approx$ (respectively: $=_{\texttt{ctx}}$) is the kernel of $\lesssim$ (respectively: $\leq_{\texttt{ctx}}$), by item (3) of Theorem 8.7. □

8.2.3 Up-to techniques

We have pointed out (Example 8.5 and 8.8) that our simulations (and bisimulations) have to be based on formal sums and cannot employ finitary reductions, as in ordinary environmental bisimulations, in order to faithfully represent contextual equivalence. However each of these features is sound and can therefore be used in proof techniques. In this section we show examples of such techniques. These techniques are very limited and we leave for future work the development of more conclusive ones.

Finitary reductions — the possibility of stopping the evaluation of a term after a few β-reductions — are interesting in enhancements with up-to context (the ability of isolating and removing common contexts in derivative terms) because sometimes such common contexts appear in the middle of a reduction. For applicability, up-to context is usually combined with further up-to techniques that allow us to bring up the common contexts. In the first up-to technique, where the coinduction game still uses formal sums, we combine up-to context with up-to lifting, so to be able to decompose related formal sums into pieces with different common contexts. In the technique, the context closure of the up-to context is only applied onto λ-terms. The closure could probably be made more powerful by applying it also on formal sums, at the price of a more complex proof, but its usefulness is unclear.

In clause (2b) below, and in the remainder of the chapter, we use the function `dirac` that takes a set of pairs of λ-terms (M, N) and returns the pairs of their (Dirac) formal sums $(1; M, 1; N)$.

Definition 8.23. A PE relation $\mathcal{R}$ is a *finite-step simulation up-to lifting and context* if:

1. $M \,\mathcal{R}\, N$ and $M \Longmapsto Y$ imply $Y \,\texttt{lift}(\mathcal{R}_{(M,N)})\, [\![N]\!]$;
2. $\sum_i p_i; \lambda x.M_i \;\mathcal{R}_{\mathcal{E}}\; \sum_j q_j; \lambda x.N_j$ implies:
   (a) $\sum_i p_i \le \sum_j q_j$;
   (b) for all $(P, Q) \in \mathcal{E}^\star$, one of the following holds:
   - there are F, G such that $\sum_i p_i; M_i\{P/x\} \Longrightarrow F$ and $\sum_j q_j; N_j\{Q/x\} \Longrightarrow G$ with $F \,\texttt{lift}(\texttt{dirac}(\mathcal{E}^\star))\, G$;
   - if $\sum_i p_i; M_i\{P/x\} \Longmapsto Y$ then $Y \;\texttt{lift}(\,\texttt{dirac}(\mathcal{E}^\star) \cup \mathcal{R}_{\mathcal{E}}\,)\; \sum_j q_j{\cdot}[\![N_j\{Q/x\}]\!]$.

The following lemma proves the soundness of the up-to lifting and context technique.

Lemma 8.24. *If $\mathcal{R}$ is a finite-step simulation up-to lifting and context then $\mathcal{R} \subseteq \lesssim_{\mathrm{fin}}$.*

Proof. Let $\mathcal{R}$ be a finite-step simulation up-to lifting and context. We prove that

$$\begin{aligned}
\mathcal{R}' &\overset{\text{def}}{=} \texttt{Pairs}(\mathcal{R}) \\
&\cup \{((M,N), Y, Z) \mid Y' \,\mathcal{R}_{(M,N)} Z \text{ and } Y \le_{\texttt{apx}} Y', \text{ for some } Y'\} \\
&\cup \{((M,N), 1; \lambda x.C[M], 1; \lambda x.C[N]) \mid M \;\mathcal{R}\; N\} \\
&\cup \{((M,N), \emptyset, Z) \mid \text{ for some } M, N, Z\}
\end{aligned}$$

is a finite-step simulation up-to lifting, from which the result follows by $\mathcal{R} \subseteq \mathcal{R}'$. The details of the proof can be found in Section 8.5.

□

Example 8.25. The up-to lifting and context technique allows us to prove that terms A, B defined in (8.1) in Section 8.1 are bisimilar. We prove $A \lesssim B$ using the PE relation

$$\{(A,B), ((A,B), [\![A]\!], [\![B]\!])\}\ .$$

Indeed, $[\![A]\!] = 1; A$ and $[\![B]\!] = \frac{1}{2}; \lambda x.x + \frac{1}{2}; \lambda x.\Omega$ and, for any pair of arguments of the form $(C[A], C[B])$ used to test the formal sums, we have $1; C[A] \oplus \Omega \longrightarrow \frac{1}{2}; C[A] + \frac{1}{2}; \Omega$ and the pair $(\frac{1}{2}; C[A] + \frac{1}{2}; \Omega, \frac{1}{2}; C[B] + \frac{1}{2}; \Omega)$ is in $\mathtt{lift}(\mathtt{dirac}(\{(A,B)\}^\star))$. Analogously, using the relation $\{(B,A), ((B,A), [\![B]\!], [\![A]\!])\} \cup \{((B,A), \emptyset, Z) \mid \text{for any } Z\}$ we prove $B \lesssim A$.

In the second up-to technique, the game is entirely played on terms, without appeal to formal sums. We present the technique in combination with forms of up-to context, up-to distribution, up-to reduction, and up-to lifting. This technique will allow us to prove the equivalence of two probabilistic fixed-point combinators in Section 8.2.4.

A *term relation* is a relation $\mathcal{T}_{(M,N)}$ on values of $\Lambda_\oplus$ and the index (M,N) is a pair of $\Lambda_\oplus$-terms. The index corresponds, intuitively, to a static environment of an environmental bisimulation. We use the notation $\mathcal{T}^{\star-}_{(M,N)}$ for $\mathcal{T}_{(M,N)} \cup \{(M,N)\}^\star$.

A term M deterministically reduces to G (notation: $\overset{\mathrm{d}}{\Longrightarrow}$) if $M \Longrightarrow G$ and only the last reduction in the sequence may be derived using rule Sum. We write $M \rhd M'$ if M and M' deterministically reduce to the same formal sum, but M' takes fewer steps. That is, there are G, m, m' with $m \geq m'$ and with $M \overset{\mathrm{d}}{\Longrightarrow}_m G$, and $M' \overset{\mathrm{d}}{\Longrightarrow}_{m'} G$. Thus, in Definition 8.26, $\rhd^\star \mathcal{T}^{\star-}_{(M,N)}$ is the set

$$\{(P,Q) \mid P \rhd^\star P' \text{ for } P' \text{ with } P'(\mathcal{T}_{(M,N)} \cup \{(M,N)\}^\star)Q\}\ .$$

We then write $F =_{\mathtt{dis}} F'$ if F and F' represent the same probability distribution. In the up-to technique below, $\rhd$ gives us the 'up-to reduction', and $=_{\mathtt{dis}}$ the 'up-to distribution'. We use up-to distribution to manipulate formal sums, which are purely syntactic objects. Finally, $\overset{\mathrm{d}}{\Longrightarrow}=_{\mathtt{dis}}$ is the composition of the two relations, i.e., $M \overset{\mathrm{d}}{\Longrightarrow}=_{\mathtt{dis}} F$ if there is F' with $M \overset{\mathrm{d}}{\Longrightarrow} F'$ and $F' =_{\mathtt{dis}} F$.

Definition 8.26. A term relation $\mathcal{T}_{(M,N)}$ is a *bisimulation up-to context closure, distribution, reduction, and lifting* if

1. $[\![M]\!] =_{\mathtt{dis}} \mathtt{dirac}(\mathcal{T}_{(M,N)}) =_{\mathtt{dis}} [\![N]\!]$;
2. if $\lambda x.M'\ \mathcal{T}_{(M,N)}\ \lambda x.N'$ then for all $(P,Q) \in \{(M,N)\}^\star$,

$$M'\{P/x\} \overset{\mathrm{d}}{\Longrightarrow}=_{\mathtt{dis}} \mathtt{lift}(\mathtt{dirac}(\rhd^\star \mathcal{T}^{\star-}_{(M,N)})) =_{\mathtt{dis}}\overset{\mathrm{d}}{\Longleftarrow} N'\{Q/x\}\ .$$

We first establish the soundness of the up-to distribution and lifting technique. For a relation $\mathcal{R}$ on formal sums, we write $\mathtt{dislift}(\mathcal{R})$ for the set of pairs F, G such that $F =_{\mathtt{dis}} \mathtt{lift}(\mathcal{R}) =_{\mathtt{dis}} G$. The up-to distribution and lifting technique is obtained by substituting $\mathtt{lift}(\cdot)$ with $\mathtt{dislift}(\cdot)$ in Definition 8.14.

Lemma 8.27. *If $\mathcal{R}$ is a finite-step simulation up-to distribution and lifting then $\mathcal{R} \subseteq \lesssim_{\mathrm{fin}}$.*

The proof follows as the one for the up-to lifting technique (Lemma 8.15), and exploits the fact that $\mathtt{dislift}(\mathtt{dislift}(\cdot)) = \mathtt{dislift}(\cdot)$. Then we derive the soundness of the full technique (the proof can be found in Section 8.5).

Lemma 8.28. *If $\mathcal{T}_{(M,N)}$ is a bisimulation up-to context closure, distribution, reduction, and lifting then $(M,N) \in \approx$.*

Remark 8.29. The first clause of the up-to technique in Definition 8.26 is not sound if $\mathcal{T}_{(M,N)}$ is substituted by $\mathcal{T}^{\star-}_{(M,N)}$: in this case, for any pair of values V, W, relation $\mathcal{T}_{(V,W)} \overset{\mathtt{def}}{=} \emptyset$ would satisfy the definition, since $[\![V]\!] = 1; V$, $[\![W]\!] = 1; W$ and $1; V \mathtt{dirac}(\{(V,W)\}^\star) 1; W$.

8.2.4 Fixed-point combinator example

In the reductions of this example, we write a Dirac formal sum $1; M$ as M, so to have reductions between λ-terms. We exploit the up-to technique of Definition 8.26 to prove the equivalence between two fixed-point combinators. One of the combinators is Υ:

$$\begin{array}{c} \Upsilon \overset{\mathtt{def}}{=} \lambda y.y(Dy(Dy)) \\ \text{where}\, D \overset{\mathtt{def}}{=} \lambda y.\lambda x.y(xx) \ . \end{array}$$

For any term L we have

$$\begin{array}{rrcl} & \Upsilon L & \longrightarrow & L(DL(DL)) \\ \text{and then} & DL(DL) & \longrightarrow\longrightarrow & L(DL(DL)) \ . \end{array} \tag{8.3}$$

The other combinator at any cycle can probabilistically decide whether to behave differently (i.e., as Turing's fixed-point combinator) or to turn for good into the previous Υ combinator:

$$\begin{array}{rl} & \Upsilon' \overset{\mathtt{def}}{=} D'D' \\ \text{where} & D' \overset{\mathtt{def}}{=} \lambda x.\lambda y.((y(Dy(Dy))) \oplus (y(xxy))) \ . \end{array}$$

Thus the computation of $\Upsilon' L$ will unveil, for a while, some L's while computing as Turing's combinator, and then will continue unveiling L's by computing as Υ. Indeed, for

$$\Upsilon'_1 \overset{\mathtt{def}}{=} \lambda y.((y(Dy(Dy))) \oplus (y(D'D'y))) \,,$$

we have

$$\begin{array}{l} \Upsilon' L \longrightarrow \Upsilon'_1 L \longrightarrow (L(DL(DL))) \oplus (L(D'D'L)) \\ \qquad\qquad \longrightarrow \frac{1}{2}; L(DL(DL)) + \frac{1}{2}; L(D'D'L) \ . \end{array} \tag{8.4}$$

We can establish $\Upsilon \approx \Upsilon'$ using the term relation

$$\mathcal{T}_{(\Upsilon,\Upsilon')} \overset{\mathtt{def}}{=} \{(\Upsilon, \Upsilon'_1)\} \ .$$

The interesting case is the bisimulation clause for (Υ, Υ'_1). Take any $M\ \{(\Upsilon, \Upsilon')\}^\star\ N$. By (8.3), we have $\Upsilon M \longrightarrow M(DM(DM))$, whereas by (8.4), $\Upsilon' N \overset{\mathrm{d}}{\Longrightarrow} \frac{1}{2}; N(DN(DN)) + \frac{1}{2}; N(D'D'N)$. Now we could conclude, up-to context closure, distribution, reduction, and lifting, if we can show that the pairs

$$\begin{array}{c} (M(DM(DM)), N(DN(DN))) \\ \text{and}\ \ (M(DM(DM)), N(\Upsilon' N)) \end{array}$$

are in $\rhd^\star \mathcal{T}^{\star-}_{(M,N)}$ This holds because: the first pair is in $\{(\Upsilon, \Upsilon')\}^\star$; for the second pair, by (8.3) we deduce $DM(DM) \rhd \Upsilon M$, and then we have

$$M(DM(DM)) \rhd^\star M(\Upsilon M)\ \{(\Upsilon, \Upsilon')\}^\star\ N(\Upsilon' N).$$

The example also shows the usefulness of static environments (whose terms need not be values) for context closures in 'up-to context' techniques.

8.3 Probabilistic call-by-value λ-calculus

In call-by-value, the static environments are not anymore sufficient. As in ordinary environmental bisimulations, we need a dynamic environment to record the values produced during the bisimulation game. In ordinary environmental bisimulations we can see such environments as tuples of values.[11] In the probabilistic case formal sums come into the picture. *Environment formal sums* are terms of the form

$$\textstyle\sum_i p_i; \widetilde{V_i}$$

(i.e., sums of weighted tuples) in which all tuples $\widetilde{V_i}$ have the same length and, as for ordinary formal sums, $0 < p_i \leq 1$ for each i and $\sum_i p_i \leq 1$. We call the length of the tuples $\widetilde{V_i}$'s the *length* of the environment formal sum. The tuples $\widetilde{V_i}$ represent the dynamic environment: the knowledge that an observer has accumulated during the bisimulation game. There may be several such elements $\widetilde{V_i}$, reflecting the possible worlds produced by the probabilistic evaluation. During the bisimulation game, the environment formal sum is updated. Viewing the environment formal sum as a matrix, in which $\widetilde{V_i}$ represents the i-row and the elements $(\widetilde{V_1})_r, (\widetilde{V_2})_r, \ldots$ (the r-th element of each row) represent the r-th column, a column is a set of values that the various possible worlds have produced at the same step of the bisimulation game. (This explains why the tuples $\widetilde{V_i}$'s of the sum have the same length.)

More precisely, in the bisimulation game at each possible world i a term M_i (constructed from the $\widetilde{V_i}$'s using a context closure discussed below) is evaluated. The evaluation of M_i yields, probabilistically, a multiset of values (as a formal sum). This multiset is empty when all evaluations from M_i diverge; in this case the whole row i disappears, meaning that in the i-th possible world the observer never receives an answer. When the multiset is non-empty, the row i is split into as many possible worlds as the values in the multiset. For instance if the evaluation of M_i produces V with probability $\frac{1}{2}$ and V' with probability $\frac{1}{3}$ then the row $p_i; \widetilde{V_i}$ is split into the two rows $\frac{1}{2} \cdot p_i; \widetilde{V_i}, V$ and $\frac{1}{3} \cdot p_i; \widetilde{V_i}, V'$.

This splitting operation is captured by the following multiplication of an environment formal sum $\sum_{i \in I} p_i; \widetilde{V_i}$ and a tuple of formal sums $Y_i = \sum_{j \in J_i} p_{i,j}; V_{i,j}$:

$$\textstyle\sum_{i \in I} p_i; \widetilde{V_i} \cdot Y_i \overset{\mathtt{def}}{=} \sum_{i \in I, j \in J_i} p_i \cdot p_{i,j}; \widetilde{V_i}, V_{i,j} \ .$$

We use $\mathbf{Y}, \mathbf{Z}$ to range over environment formal sums, and we sometimes treat a formal sum as a special case of environment formal sum in which all tuples have length one.

The view of environment formal sums as matrices is illustrated in Figure 8.2, for an environment formal sum $\mathbf{Y} \overset{\mathtt{def}}{=} \sum_{1 \leq i \leq 3} p_i; \widetilde{V_i}$ of length 4. The figure also illustrates the extraction of the column r of the formal sum, written $\mathbf{Y}\lfloor_r$, that yields the tuple of values along the same column, and the multiplication of an environment formal sum with formal sums resulting from the semantics of terms, one per row (where $\mathsf{I} = \lambda x.x$ is the identity function).

The *dynamic environment* of two environment formal sums $\sum_{i \in I} p_i; \widetilde{V_i}$ and $\sum_{j \in J} q_j; \widetilde{W_j}$ is the pair of tuples (of tuples) $(\{\widetilde{V_i}\}_{i \in I}, \{\widetilde{W_j}\}_{j \in J})$. In environmental bisimulations, the

[11] The definition of environmental bisimulation given in Chapter 6 (Definition 6.7) for non-probabilistic calculi has a unique notion of environment $\mathcal{E}$, which corresponds to what we call here the dynamic environment. In remark 6.8, we have seen how this environment can be alternatively formalized in non-probabilistic calculi using tuples of values.

$$\mathbf{Y} = \sum \begin{array}{r|c|c|c|c|l}
 & \overbrace{\phantom{V_{1,1}}}^{\mathbf{Y}{\downharpoonleft_1}} & \overbrace{\phantom{V_{1,2}}}^{\mathbf{Y}{\downharpoonleft_2}} & \overbrace{\phantom{V_{1,3}}}^{\mathbf{Y}{\downharpoonleft_3}} & \overbrace{\phantom{V_{1,4}}}^{\mathbf{Y}{\downharpoonleft_4}} & \\
\cline{2-5}
p_1; & V_{1,1} & V_{1,2} & V_{1,3} & V_{1,4} & \} \; \widetilde{V_1} \\
\cline{2-5}
p_2; & V_{2,1} & V_{2,2} & V_{2,3} & V_{2,4} & \} \; \widetilde{V_2} \\
\cline{2-5}
p_3; & V_{3,1} & V_{3,2} & V_{3,3} & V_{3,4} & \} \; \widetilde{V_3} \\
\cline{2-5}
\end{array}$$

$$\sum \begin{array}{r|c|c|l}
\cline{2-3}
p_1; & V_{1,1} & V_{1,2} & \cdot [\![I \oplus \Omega]\!] \\
\cline{2-3}
p_2; & V_{2,1} & V_{2,2} & \cdot [\![I \oplus \lambda.\Omega]\!] \\
\cline{2-3}
\end{array}
= \sum \begin{array}{r|c|c|c|}
\cline{2-4}
\frac{p_1}{2}; & V_{1,1} & V_{1,2} & \mathsf{I} \\
\cline{2-4}
\frac{p_2}{2}; & V_{2,1} & V_{2,2} & \mathsf{I} \\
\cline{2-4}
\frac{p_2}{2}; & V_{2,1} & V_{2,2} & \lambda.\Omega \\
\cline{2-4}
\end{array}$$

Figure 8.2: Formal sums as matrices

input for two higher-order functions is constructed as the context closure of their environments. In call-by-value, the environments have both a static and a dynamic component and the inputs are constructed accordingly. Given a static environment (M, N) and a dynamic environment $(\{\widetilde{V_i}\}_i, \{\widetilde{W_j}\}_j)$, their context closure, written

$$(\{M, \widetilde{V_i}\}_i, \{N, \widetilde{W_j}\}_j)^{\widehat{\star}}$$

is the set of all pairs of tuples $(\{T_i\}_i, \{U_j\}_j)$ for which there is a context C such that for every i we have $T_i = C[M, \widetilde{V_i}]$, and for every j we have $U_j = C[N, \widetilde{W_j}]$. Thus every T_i is obtained from the same context C by filling its holes with the first element M of the static environment and the dynamic environment $\widetilde{V_i}$. Similarly for U_j, using N, the tuple $\widetilde{W_j}$ and the same context C. Moreover, as we are in call-by-value, C should be a value context, that is, terms T_i and U_j are values for all i, j.

The operational semantics of call-by-value is defined as in call-by-name, provided that the rule for β-reduction and the evaluation contexts are redefined thus:

$$\textsc{BetaV}\ \frac{}{(\lambda x.M)V \longrightarrow 1; M\{{}^{V}\!/\!{}_{x}\}}$$

$$\text{Evaluation contexts} \qquad C = [\cdot] \;\Big|\; CM \;\Big|\; VC$$

8.3.1 Environmental bisimulation

In call-by-value, a probabilistic environmental relation (that we still abbreviate as PE relation) is like for call-by-name, except that formal sums are replaced by environment formal sums. That is, each element of the relation is either of the form (M, N) (a pair of $\Lambda_\oplus$-terms) or $\mathbf{Y} \mathrel{\mathcal{R}_\mathcal{E}} \mathbf{Z}$ (two environment formal sums, collecting the dynamic environment, with a static environment).
If $\mathcal{E} = (M, N)$ is a static environment, then $\mathcal{E}_1$ and $\mathcal{E}_2$ denote the projections, i.e., the terms M and N, respectively.

In a PE relation, related environment formal sums are *compatible*, meaning that they have the same length. In the remainder, compatibility of environment formal sums is tacitly assumed.

Definition 8.30 (Environmental bisimulation, call-by-value)**.** A PE relation is a *(PE) bisimulation* if

1. $M \mathrel{\mathcal{R}} N$ implies $[\![M]\!] \mathrel{\mathcal{R}_{(M,N)}} [\![N]\!]$;
2. $\sum_i p_i; \widetilde{V_i} \mathrel{\mathcal{R}_\mathcal{E}} \sum_j q_j; \widetilde{W_j}$ implies:
   (a) $\sum_i p_i = \sum_j q_j$;
   (b) for all r, if $(\widetilde{V_i})_r = \lambda x.M_i$ and $(\widetilde{W_j})_r = \lambda x.N_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\widehat{\star}}$ we have
   $$\textstyle\sum_i p_i; \widetilde{V_i} \cdot [\![M_i\{{}^{T_i}\!/\!{}_x\}]\!] \mathrel{\mathcal{R}_\mathcal{E}} \sum_j q_j; \widetilde{W_j} \cdot [\![N_j\{{}^{U_j}\!/\!{}_x\}]\!] \;;$$
   (c) $\sum_i p_i; \widetilde{V_i} \cdot [\![\mathcal{E}_1]\!] \mathrel{\mathcal{R}_\mathcal{E}} \sum_j q_j; \widetilde{W_j} \cdot [\![\mathcal{E}_2]\!]$.

(PE) bisimilarity, $\approx$, is the union of all PE bisimulations, and the corresponding *similarity* is $\lesssim$.

The structure of the above definition is similar to that of ordinary environmental bisimulations. There are three main differences: first, the appearance of formal sums and of probability measures (notably in clause (2a)); second, the use of an (infinitary) big-step semantics, rather than a small-step, which shows up in the function $[\![\ ]\!]$ in clauses (1), (2b) and (2c); thirdly the appearance of a static environment, that is used in the context closure and in clauses (1) and (2c). In clause (2b), the related environment formal sums, viewed as matrices, grow by the addition of a new column resulting, on left-hand side, from the multiplication of each row $p_i; \widetilde{V_i}$ with the formal sum $[\![M_i\{{}^{T_i}\!/\!{}_x\}]\!]$, and similarly on the right-hand side. Thus the compatibility between related environment formal sums is maintained. Clause (2c) allows to re-evaluate the static environment at any time. This clause and other features are necessary in order to achieve full abstraction in the imperative case (see Section 8.4.1); they could be removed in pure call-by-value, following [CD14].

Remark 8.31. Clause (1) could be substituted by

(1′) $M \mathrel{\mathcal{R}} N$ implies $1; \emptyset \mathrel{\mathcal{R}_{(M,N)}} 1; \emptyset$

where $1; \emptyset$ is the Dirac formal sum with empty environment. Clause (2c) then guarantees that $[\![M]\!] \mathrel{\mathcal{R}_{(M,N)}} [\![N]\!]$. We did not use this definition for continuity with the call-by-name case, and since not needed for pure calculi. By contrast, a modification of clause (1) analogous to (1′) is used in Definition 8.41 of bisimulation for imperative calculi (see Example 8.42).

Theorem 8.32.

1. *$\approx$ and $\lesssim$ are the largest bisimulation and simulation, respectively.*
2. *$\lesssim$ is a preorder, and $\approx$ an equivalence.*
3. $\approx = \lesssim \cap \lesssim^{-1}$.

Proof. The proof is analogous to the one for call-by-name. The addition of the dynamic environment is not relevant for this proof, and the extra clause (2c) is treated analogously to clause (2b). □

Example 8.33. Terms M and N in Example 8.5 are equivalent in call-by-name, but not in call-by-value. A bisimulation relating these terms should contain the formal sums $[\![M]\!] = \frac{1}{2}; \lambda.\lambda.\Omega + \frac{1}{2}; \lambda.\Omega$ and $[\![N]\!] = 1; N$, with static environment $\mathcal{E} = (M, N)$, and thus the triple $(\mathcal{E}, \frac{1}{2}; \lambda.\lambda.\Omega, \lambda.\Omega, \frac{1}{2}; N, \lambda.\Omega)$ would be in the relation as well. However, the values in the first column of the dynamic environment can be tested again, by clause (2b) of bisimulation, leading to the triple

$$(\mathcal{E}, \tfrac{1}{2}; \lambda.\lambda.\Omega, \lambda.\Omega, \lambda.\Omega, \tfrac{1}{4}; N, \lambda.\Omega, \lambda.\Omega)\,,$$

which does not satisfy clause (2a).

The main results for environmental bisimilarity and similarity in call-by-value (congruence and full abstraction with respect to contextual preorder and equivalence) are as for call-by-name, and the structure of the proofs is similar. The details are however different due to the presence of dynamic environments. As for call-by-name, so in call-by-value to reason about bisimilarity and similarity we need a finite-step simulation, with challenges produced by the finitary big-step approximants. To make sure that the challenges are finite-step, we define *extended environment formal sums*, i.e., terms

$$\textstyle\sum_i p_i; \widetilde{V_i}; M_i$$

in which the environment formal sum $\sum_i p_i; \widetilde{V_i}$ is extended with an additional column of arbitrary $\Lambda_\oplus$-term (not necessarily values). Intuitively, an element $\widetilde{V_i}; M_i$ indicates that the λ-term M_i has to be run with an observer whose knowledge is $\widetilde{V_i}$. Extended environment formal sums are ranged over by $\mathbf{F}, \mathbf{G}$ and $\mathtt{val}(\mathbf{F})$ is defined analogously to formal sums:

$$\mathtt{val}(\textstyle\sum_i p_i; \widetilde{V_i}; M_i) \overset{\mathtt{def}}{=} \sum_{\{i \mid M_i \text{is a value}\}} p_i; \widetilde{V_i}, M_i\ .$$

Extended environment formal sums allow us to define the multi-step reduction relation from extended environment formal sums to environment formal sums: for $\mathbf{F} = \sum_{i \in I} p_i; \widetilde{V_i}; M_i + \mathbf{G}$, where I is a finite set, we set

$$\frac{M_i \Longmapsto Y_i \quad \text{for every } i}{\mathbf{F} \Longmapsto \sum_{i \in I} p_i; \widetilde{V_i} \cdot Y_i + \mathtt{val}(\mathbf{G})}$$

This intuitively corresponds to the multi-step reduction relation from formal sums to value formal sums. In clause (1) below we see formal sums as special cases of environment formal sums. For an extended environment formal sum $\mathbf{F} = \sum_i p_i; \widetilde{V_i}; M_i$, we let $[\![\mathbf{F}]\!] \overset{\mathtt{def}}{=} \sup\{\mathbf{Y} \mid \mathbf{F} \Longmapsto \mathbf{Y}\}$. Since $\sum_i p_i; \widetilde{V_i} \cdot [\![M_i]\!] = [\![F]\!]$, we could have equivalently defined the operational semantics for call-by-value directly on extended environment formal sums.

Additional notation. We extend to (extended) environment formal sums the notations for β-reduction, contexts, and application on formal sums, and introduce a notation for extending the dynamic environments. We use r to range over indexes of columns of environment formal sums, and we let $\mid \mathbf{Y} \mid$ denote the length of an environment formal sum $\mathbf{Y}$. Abusing notation, we sometimes write $\mid \mathbf{Y} \mid$ also for the index set $\{1, ..., \mid \mathbf{Y} \mid\}$. Let $\mathbf{Y} = \sum_i p_i; \widetilde{V_i}$ and $\mathbf{F} = \sum_i p_i; \widetilde{V_i}; M_i$ and P be a term.

- for any r, if $(\widetilde{V_i})_r = \lambda x.M_i$ we let $\mathbf{Y}; \mathbf{Y}\lfloor_r \bullet P \overset{\mathtt{def}}{=} \sum_i p_i; \widetilde{V_i}; M_i\{P/x\}$;

- for any r, if $(\widetilde{V_i})_r = \lambda x.M_i$ and C is a context with holes with indexes ranging over $\mid \mathbf{Y} \mid +1$ we let $\mathbf{Y};\mathbf{Y}\lfloor_r \bullet C[P,\mathbf{Y}] \overset{\mathtt{def}}{=} \sum_i p_i;\widetilde{V_i};M_i\{C[P,\widetilde{V_i}]/x\}$;
- $\mathbf{Y};P \overset{\mathtt{def}}{=} \sum_i p_i;\widetilde{V_i};P$;
- for any context C, we let $C[\mathbf{F}] \overset{\mathtt{def}}{=} \sum_i p_i;\widetilde{V_i};C[M_i]$;
- for $F_j = \sum_{i\in I_j} p_{j,i};M_{j,i}$, we let $\sum_j q_j;C_j[F_j] \overset{\mathtt{def}}{=} \sum_{j,i\in I_j} q_j \cdot p_{i,j};C_j[M_{j,i}]$.

Using this notation, the call-by-value bisimulation clauses for environment formal sums become as follows:

(2) $\mathbf{Y} \mathrel{\mathcal{R}_{\mathcal{E}}} \mathbf{Z}$ implies:

  (a) $\mathtt{weight}(\mathbf{Y}) = \mathtt{weight}(\mathbf{Z})$;

  (b) for all r and for all contexts C, $[\![\mathbf{Y};\mathbf{Y}\lfloor_r \bullet C[\mathcal{E}_1,\mathbf{Y}]]\!] \mathrel{\mathcal{R}_{\mathcal{E}}} [\![\mathbf{Z};\mathbf{Z}\lfloor_r \bullet C[\mathcal{E}_2,\mathbf{Z}]]\!]$;

  (c) $[\![\mathbf{Y};\mathcal{E}_1]\!] \mathrel{\mathcal{R}_{\mathcal{E}}} [\![\mathbf{Z};\mathcal{E}_2]\!]$.

As for call-by-name, we will use this additional notation only for proofs.

Definition 8.34. A PE relation is a *finite-step simulation* if

1. $M \mathrel{\mathcal{R}} N$ and $M \Longmapsto Y$ imply $Y \mathrel{\mathcal{R}_{(M,N)}} [\![N]\!]$;
2. $\sum_i p_i;\widetilde{V_i} \mathrel{\mathcal{R}_{\mathcal{E}}} \sum_j q_j;\widetilde{W_j}$ implies:
   (a) $\sum_i p_i \leq \sum_j q_j$;
   (b) for all r, if $(\widetilde{V_i})_r = \lambda x.M_i$ and $(\widetilde{W_j})_r = \lambda x.N_j$ then for all $(\{T_i\}_i,\{U_j\}_j) \in (\{\mathcal{E}_1,\widetilde{V_i}\}_i,\{\mathcal{E}_2,\widetilde{W_j}\}_j)^{\widehat{\star}}$ we have $\sum_i p_i;\widetilde{V_i};M_i\{T_i/x\} \Longmapsto \mathbf{Y}$ implies
   $\mathbf{Y} \mathrel{\mathcal{R}_{\mathcal{E}}} \sum_j q_j;\widetilde{W_j} \cdot [\![N_j\{U_j/x\}]\!]$;
   (c) $\sum_i p_i;\widetilde{V_i};\mathcal{E}_1 \Longmapsto \mathbf{Y}$ implies $\mathbf{Y} \mathrel{\mathcal{R}_{\mathcal{E}}} \sum_j q_j;\widetilde{W_j} \cdot [\![\mathcal{E}_2]\!]$.

We write $\lesssim_{\text{fin}}$ for the union of all finite-step simulations. Analogously to call-by-name, we use a saturation by approximants and a saturation by suprema to move from a simulation to a finite-step simulation and conversely, and exploit this to prove that similarity and finite-step similarity coincide.

Lemma 8.35. *Relations $\lesssim$ and $\lesssim_{\text{fin}}$ coincide.*

The proof follows as in call-by-name. We prove that the saturation by approximants of a simulation is a finite-step simulation and that the saturation by suprema of a finite-step simulation is a simulation. Clause (2c) is treated analogously to clause (2b).

As in call-by-name, we derive congruence for bisimilarity and similarity by first proving the property for finite-step similarity. We also exploit a combination of two up-to techniques for finite-step simulation, namely up-to lifting and up-to environment. Up-to lifting is defined analogously to call-by-name, using the lifting operation on environment formal sums. Up-to environment intuitively allows us to exchange columns of environment formal sums (when these are viewed as matrices as in Figure 8.2), and to add new columns. Adding columns is safe because it means enlarging the dynamic environment: terms that are equal in the larger environment are also so in the smaller environment as the tests that can be built (when playing the bi-simulation game) with the latter environment are a subset of those that are obtained from the former environment.

The up-to environment technique is based on the definition of the preorder $\leq_{\texttt{env}}$ on pairs of formal sums (where, for each pair, the formal sums in the pair have the same length). Environment formal sums $(\mathbf{Y}, \mathbf{Z})$ are below formal sums $(\mathbf{Y}', \mathbf{Z}')$ if, in the second pair, the dynamic environment of the first pair is extended and the columns have been possibly permuted. Formally, $(\mathbf{Y}, \mathbf{Z}) \leq_{\texttt{env}} (\mathbf{Y}', \mathbf{Z}')$ if $\mathbf{Y} = \sum_i p_i; \widetilde{V_i}, \mathbf{Z} = \sum_j q_j; \widetilde{W_j}, \mathbf{Y}' = \sum_i p_i; \widetilde{V_i'}, \mathbf{Z}' = \sum_j q_j; \widetilde{W_j'}$ and for every index r in $\mid \mathbf{Y} \mid$ there is an index r' in $\mid \mathbf{Y}' \mid$ such that $\mathbf{Y}\lfloor_r = \mathbf{Y}'\lfloor_{r'}$ and $\mathbf{Z}\lfloor_r = \mathbf{Z}'\lfloor_{r'}$.
Hence, given a relation $\mathcal{R}$ on environment formal sums, $\mathbf{Y}\,\texttt{lift}(\geq_{\texttt{env}}(\mathcal{R}))\,\mathbf{Z}$ holds if there are p_i, $\mathbf{Y}_i$ and $\mathbf{Z}_i$, for i ranging over some index set, such that $\mathbf{Y} = \sum_i p_i \cdot \mathbf{Y}_i$ and $\mathbf{Z} = \sum_i p_i \cdot \mathbf{Z}_i$, and for every i there are $\mathbf{Y}_i', \mathbf{Z}_i'$ such that $\mathbf{Y}_i' \mathcal{R} \mathbf{Z}_i'$ and $(\mathbf{Y}_i, \mathbf{Z}_i) \leq_{\texttt{env}} (\mathbf{Y}_i', \mathbf{Z}_i')$.

Definition 8.36. A PE relation is a *probabilistic finite-step simulation up-to lifting and environment* if:

1. $M \mathcal{R} N$ implies $M \Longmapsto Y$ implies $Y\,\texttt{lift}(\geq_{\texttt{env}}(\mathcal{R}_{(M,N)}))\,[\![N]\!]$;
2. $\sum_i p_i; \widetilde{V_i}\,\mathcal{R}_{\mathcal{E}} \sum_j q_j; \widetilde{W_j}$ implies:
   (a) $\sum_i p_i \leq \sum_i q_i$;
   (b) for all r, if $(\widetilde{V_i})_r = \lambda x.M_i$ and $(\widetilde{W_j})_r = \lambda x.N_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\star}$ we have that $\sum_i p_i; \widetilde{V_i}; M_i\{T_i/x\} \Longmapsto \mathbf{Y}$ implies
   $$\mathbf{Y}\,\texttt{lift}(\geq_{\texttt{env}}(\mathcal{R}_{\mathcal{E}}))\,\textstyle\sum_j q_j; \widetilde{W_j} \cdot [\![N_j\{U_j/x\}]\!] ;$$
   (c) $\sum_i p_i; \widetilde{V_i}; \mathcal{E}_1 \Longmapsto \mathbf{Y}$ implies $\mathbf{Y}\,\texttt{lift}(\geq_{\texttt{env}}(\mathcal{R}_{\mathcal{E}}))\,\sum_j q_j; \widetilde{W_j} \cdot [\![\mathcal{E}_2]\!]$.

We now prove that the up-to lifting and environment technique is sound.

Theorem 8.37. *If $\mathcal{R}$ is a finite-step simulation up-to lifting and environment then $\mathcal{R} \subseteq \lesssim_{\text{fin}}$.*

Proof. Let $\mathcal{R}$ be a finite-step simulation up-to lifting and environment. Then the following is a finite-step simulation:

$$\mathcal{S} = \texttt{Pairs}(\mathcal{R}) \cup \textstyle\bigcup_{\mathcal{E}} \texttt{lift}(\geq_{\texttt{env}}(\mathcal{R}_{\mathcal{E}}))$$

If $M\,\mathcal{S}\,N$ then $M \mathcal{R} N$, which implies that if $M \Longmapsto \mathbf{Y}$ then $\mathbf{Y}\,\texttt{lift}(\geq_{\texttt{env}}(\mathcal{R}_{(M,N)}))\,[\![N]\!]$. Hence, $\mathbf{Y}\,\mathcal{S}_{(M,N)}\,[\![N]\!]$.
Let $\mathbf{Y}\,\mathcal{S}_{(M,N)}\,\mathbf{Z}$, i.e., $\mathbf{Y} = \sum_i p_i \cdot \mathbf{Y}_i$, $\mathbf{Z} = \sum_i p_i \cdot \mathbf{Z}_i$ and for every i there are $\mathbf{Y}_i'\,\mathcal{R}_{(M,N)}\,\mathbf{Z}_i'$ such that $(\mathbf{Y}_i, \mathbf{Z}_i) \leq_{\texttt{env}} (\mathbf{Y}_i', \mathbf{Z}_i')$.
Clause (a) holds since for every i, $\texttt{weight}(\mathbf{Y}_i') \leq \texttt{weight}(\mathbf{Z}_i')$. Therefore, $\texttt{weight}(\mathbf{Y}) \leq \texttt{weight}(\mathbf{Z})$.
To prove clause (b), suppose that $\mathbf{Y}; \mathbf{Y}\lfloor_r \bullet C[M, \mathbf{Y}] \Longmapsto \mathbf{W}$. Then $\mathbf{W} = \sum_i p_i \cdot \mathbf{W}_i$, for some $\mathbf{W}_i$ such that $\mathbf{Y}_i; \mathbf{Y}_i\lfloor_r \bullet C[M, \mathbf{Y}_i] \Longmapsto \mathbf{W}_i$. Analogously, we have

$$[\![\mathbf{Z}; \mathbf{Z}\lfloor_r \bullet C[N, \mathbf{Z}]]\!] = \textstyle\sum_i p_i \cdot [\![\mathbf{Z}_i; \mathbf{Z}_i\lfloor_r \bullet C[N, \mathbf{Z}_i]]\!].$$

Suppose that $\mathbf{Y}_i; \mathbf{Y}_i{\downharpoonright_r} \bullet C[M, \mathbf{Y}_i] = \sum_j q_j; \widetilde{V_j}; M_j$ and $\mathbf{Z}_i; \mathbf{Z}_i{\downharpoonright_r} \bullet C[M, \mathbf{Z}_i] = \sum_k q'_k; \widetilde{W_k}; N_k$. Then it follows from the definition of the environment preorder $\leq_{\texttt{env}}$ that there are r_i and C_i such that $\mathbf{Y}'_i; \mathbf{Y}'_i{\downharpoonright_{r_i}} \bullet C_i[M, \mathbf{Y}'_i] = \sum_j q_j; \widetilde{V'_j}; M_j$ and $\mathbf{Z}'_i; \mathbf{Z}'_i{\downharpoonright_{r_i}} \bullet C_i[N, \mathbf{Z}'_i] = \sum_k q'_k; \widetilde{W'_k}; N_k$. Therefore, there is a $\mathbf{W}'_i$ such that $\mathbf{Y}'_i; \mathbf{Y}'_i{\downharpoonright_{r_i}} \bullet C_i[M, \mathbf{Y}'_i] \Longmapsto \mathbf{W}'_i$ and

$$(\mathbf{W}_i, [\![\mathbf{Z}_i; \mathbf{Z}_i{\downharpoonright_r} \bullet C[N, \mathbf{Z}_i]]\!]) \leq_{\texttt{env}} (\mathbf{W}'_i, [\![\mathbf{Z}'_i; \mathbf{Z}'_i{\downharpoonright_{r_i}} \bullet C_i[N, \mathbf{Z}'_i]]\!])$$

with $\mathbf{W}'_i\,\texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{(M,N)}))\, [\![\mathbf{Z}'_i; \mathbf{Z}'_i{\downharpoonright_{r_i}} \bullet C_i[N, \mathbf{Z}'_i]]\!])$, since $\mathcal{R}$ is a finite-step simulation up-to lifting and environment and $\mathbf{Y}'_i\ \mathcal{R}_{(M,N)}\ \mathbf{Z}'_i$. Hence, we have

$$\mathbf{W}\,\texttt{lift}(\geq_{\texttt{env}} (\,\texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{(M,N)}))\,))\, [\![\mathbf{Z}; \mathbf{Z}{\downharpoonright_r} \bullet C[N, \mathbf{Z}]]\!]$$

and the result follows from $\texttt{lift}(\geq_{\texttt{env}} (\,\texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{(M,N)}))\,)) = \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{(M,N)}))$. Finally, clause (c) follows analogously to clause (b).

□

Having these up-to techniques, we derive the result by showing that the context closure (which, differently from call-by-name, is now applied both to terms and to the environments of formal sums) of a finite-step simulation saturated by approximants is a finite-step simulation up-to lifting and environment.

Lemma 8.38. *If $M \lesssim_{\text{fin}} N$ then for every context C we have that $C[M] \lesssim_{\text{fin}} C[N]$.*

Proof. We first define, for any pairs of terms (M, N), the preorder $\leq_{\texttt{cce}(M,N)}$ (context closure of environments) on pairs of environment formal sums: $(\mathbf{Y}, \mathbf{Z}) \leq_{\texttt{cce}(M,N)} (\mathbf{Y}', \mathbf{Z}')$ if $\mathbf{Y} = \sum_i p_i; \widetilde{V_i}$, $\mathbf{Z} = \sum_j q_j; \widetilde{W_j}$ with $\mid \mathbf{Y} \mid = \mid \mathbf{Z} \mid$, $\mathbf{Y}' = \sum_i p_i; \widetilde{V'_i}$, $\mathbf{Z}' = \sum_j q_j; \widetilde{W'_j}$ with $\mid \mathbf{Y}' \mid = \mid \mathbf{Z}' \mid$ and

- for every index r in $\mid \mathbf{Y} \mid$ there is an index r' in $\mid \mathbf{Y}' \mid$ such that such that $\mathbf{Y}{\downharpoonright_r} = \mathbf{Y}'{\downharpoonright_{r'}}$ and $\mathbf{Z}{\downharpoonright_r} = \mathbf{Z}'{\downharpoonright_{r'}}$
- for every index r' in $\mid \mathbf{Y}' \mid$ there is a value-context C whose indexes range over $\mid \mathbf{Y} \mid$ such that for every i, j, $(\widetilde{V'_i})_{r'} = C[M, \widetilde{V_i}]$ and $(\widetilde{W'_j})_{r'} = C[N, \widetilde{W_j}]$ (i.e., $(\mathbf{Y}'{\downharpoonright_{r'}}, \mathbf{Z}'{\downharpoonright_{r'}}) \in (\{(M, \widetilde{V_i}\}_i, \{N, \widetilde{W_j}\}_j)^{\widehat{\star}})$.

Intuitively, this corresponds to considering all finite subsets of the context closure of a relation: given formal sums $(\mathbf{Y}, \mathbf{Z})$, related elements are columns of their environments that have the same indexes, and $(\mathbf{Y}', \mathbf{Z}')$ expands $(\mathbf{Y}, \mathbf{Z})$ (up-to permutation of columns of the pair) with columns that are obtained by filling the same context with related columns and with the static environment.

Let $\mathcal{R}$ be a finite-step simulation saturated by approximants such that $M\ \mathcal{R}\ N$. We can assume without loss of generality that this is the only pair of terms in $\mathcal{R}$ and that for any $\mathcal{E}$ such that $\mathcal{R}_{\mathcal{E}} \subseteq \mathcal{R}$ we have $\mathcal{E} = (M, N)$. We can also assume that $\mathcal{R}_{(M,N)}$ contains all pairs in relation $\{(\emptyset, \mathbf{Y}) \mid \mathbf{Y} \text{ is an environment formal sum}\}$, and that it contains the pair of Dirac formal sums with empty environment $(1; \emptyset, 1; \emptyset)$, since these pairs trivially satisfy the finite-step simulation clauses.

Let $\mathcal{R}^{\texttt{cce}}_{(M,N)} \overset{\texttt{def}}{=} \leq_{\texttt{cce}(M,N)} (\mathcal{R}_{(M,N)})$, which is turn denotes the set

$$\{(\mathbf{Y}, \mathbf{Z}) \mid \exists \mathbf{Y}', \mathbf{Z}' \text{ such that } (\mathbf{Y}', \mathbf{Z}') \leq_{\texttt{cce}(M,N)} (\mathbf{Y}, \mathbf{Z}) \wedge \mathbf{Y}'\ \mathcal{R}_{(M,N)}\ \mathbf{Z}'\}.$$

We prove that the following is a finite-step simulation up-to lifting and environment:

$$\mathcal{S} \overset{\mathsf{def}}{=} \{(C[M], C[N] \mid M \mathrel{\mathcal{R}} N\} \cup \{((C[M], C[N]), \mathbf{Y}, \mathbf{Z}) \ \mid \ \mathbf{Y} \mathrel{\mathcal{R}^{\mathsf{cce}}_{(M,N)}} \mathbf{Z}\}.$$

We require $\mathcal{R}_{(M,N)}$ to contain the pair $(1;\emptyset, 1;\emptyset)$ in order to include triples such as

$$((\lambda x.C[M], \lambda x.C[N]), 1; \lambda x.C[M], 1; \lambda x.C[N])$$

which must be in $\mathcal{S}$ since $\lambda x.C[M] \mathrel{\mathcal{S}} \lambda x.C[N]$.
We assume $\emptyset \mathrel{\mathcal{R}_{(M,N)}} \mathbf{Z}$ for any $\mathbf{Z}$ since, for any C that is not a value context, we have $C[M] \Longmapsto \emptyset$, so we want to derive $\emptyset \, \mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}^{\mathsf{cce}}_{(M,N)})) \, [\![C[N]]\!]$.
Finally, relation $\mathcal{R}$ must be saturated by approximants since, as shown in Example 8.13, a finite-step simulation need not to be and it might contain, e.g., the triple $((P,Q), \frac{1}{2}; Q + \frac{1}{4}; Q, 1; Q)$ but not the triple $((P,Q), \frac{1}{4}; Q, 1; Q)$. However, if $C = \mathsf{I}[\cdot]$ then $\frac{1}{2}; C[Q] + \frac{1}{4}; C[Q] \Longmapsto \frac{1}{4}; Q$ and it might be the case that there are no $\mathbf{Y}, \mathbf{Z}$ such that $\mathbf{Y} \mathrel{\mathcal{R}_{(P,Q)}} \mathbf{Z}$ and $(\frac{1}{4}; Q, 1; Q) \leq_{\mathtt{cce}} (\mathbf{Y}, \mathbf{Z})$.
The proof follows the same steps as the congruence proof for the imperative λ-calculus, and we thereby refer the reader to Section 8.5, proof of Theorem 8.51. □

8.3.2 Contextual equivalence

The definitions of the contextual preorder and equivalence, $\leq_{\mathtt{ctx}}$ and $=_{\mathtt{ctx}}$, are as for call-by-name.

Theorem 8.39 (Completeness). *If $M \leq_{\mathtt{ctx}} N$ then $M \lesssim N$.*

Proof. We prove that the relation

$$\begin{aligned} \mathcal{R} = \{ & ((M,N), \textstyle\sum_i p_i; V_1^i, ..., V_n^i, \sum_j q_j; W_1^j, ..., W_n^j) \mid \\ & M \leq_{\mathtt{ctx}} N \ \wedge \ \exists C \text{ such that } [\![C[M]]\!] = \textstyle\sum_i p_i; \lambda x.x V_1^i ... V_n^i \\ & \wedge \ [\![C[N]]\!] = \textstyle\sum_j q_j; \lambda x.x W_1^j ... W_n^j \} \end{aligned}$$

is a simulation. Then we derive the result as follows: let $M \leq_{\mathtt{ctx}} N$ and $P = (\lambda y \lambda x.xy)$. Then if $[\![M]\!] = \sum_i p_i; V_i$ and $[\![N]\!] = \sum_j q_j; W_j$ then $[\![PM]\!] = \sum_i p_i; \lambda x.x V_i$ and $[\![PN]\!] = \sum_j q_j; \lambda x.x W_j$, which implies that $[\![M]\!] \mathrel{\mathcal{R}_{(M,N)}} [\![N]\!]$. Hence, $M \lesssim N$.

To prove that $\mathcal{R}$ is a simulation, suppose that there are M, N, C such that $M \leq_{\mathtt{ctx}} N$, $[\![C[M]]\!] = \sum_i p_i; \lambda x.x V_1^i ... V_n^i$ and $[\![C[N]]\!] = \sum_j q_j; \lambda x.x W_1^j ... W_n^j$. Let $\mathbf{Y} = \sum_i p_i; V_1^i, ..., V_n^i$ and $\mathbf{Z} = \sum_j q_j; W_1^j, ..., W_n^j$.

We want to prove that for any $r \in \{1, ..., n\}$ and for any context C'',

$$[\![\mathbf{Y}; \mathbf{Y}\lfloor_r \bullet C''[M, \mathbf{Y}]]\!] \mathrel{\mathcal{R}_{(M,N)}} [\![\mathbf{Z}; \mathbf{Z}\lfloor_r \bullet C''[N, \mathbf{Z}]]\!]$$

which is equivalent to saying that there is a context D such that

$$\begin{aligned} [\![D[M]]\!] &= \textstyle\sum_i p_i; \lambda x.(x V_1^i ... V_n^i [\![V_r^i \bullet C''[M, V_1^i, ..., V_n^i]]\!]) \\ [\![D[N]]\!] &= \textstyle\sum_j q_j; \lambda x.(x W_1^j ... W_n^j [\![W_r^j \bullet C''[N, W_1^j, ..., W_n^j]]\!]). \end{aligned}$$

Let C' be any context and let

$$P_{M,C'} = \lambda x_1, ..., x_n.(\lambda z, x.x x_1 ... x_n z) C'[M, x_1, ..., x_n]$$

and $P_{N,C'}$ the same term with M substituted to N. It follows from $M \leq_{\mathtt{ctx}} N$ that $C[M]P_{M,C'} \leq_{\mathtt{ctx}} C[N]P_{N,C'}$. We have that:

$$\begin{aligned}
[\![C[M]P_{M,C'}]\!] &= [\![[\![C[M]]\!]P_{M,C'}]\!] \\
&= [\![\textstyle\sum_i p_i; (P_{M,C'}V_1^i...V_n^i)]\!] \\
&= [\![\textstyle\sum_i p_i; (\lambda z, x.xV_1^i...V_n^i z)C'[M, V_1^i, ..., V_n^i]]\!] \\
&= \textstyle\sum_i p_i; (\lambda z, x.xV_1^i...V_n^i z)[\![C'[M, V_1^i, ..., V_n^i]]\!] \\
&= \textstyle\sum_i p_i; (\lambda x.xV_1^i...V_n^i [\![C'[M, V_1^i, ..., V_n^i]]\!])
\end{aligned}$$

and analogously for N:

$$[\![C[N]P_{N,C'}]\!] = \textstyle\sum_j q_j; (\lambda x.xW_1^j...W_n^j [\![C'[N, W_1^j, ..., W_n^j]]\!])$$

Then by the definition of $\mathcal{R}$ we have that for any context C',

$$\textstyle\sum_i p_i; V_1^i, ..., V_n^i \cdot [\![C'[M, V_1^i, ..., V_n^i]]\!] \ \mathcal{R}_{(M,N)} \ \sum_j q_j; W_1^j, ..., W_n^j \cdot [\![C'[N, W_1^j, ..., W_n^j]]\!]$$

This holds in particular for any context of the form $C' = [\cdot]_{r+1}C''$, for $r \in \{1, ..., n\}$ and C'' a value-context, which implies the clause (b) of simulation on formal sums.
Clause (c) is proved using the same result, by taking $C' = [\cdot]_1$.
Finally, to verify the first clause of simulation on formal sums, we must prove that $\sum_i p_i \leq \sum_j q_j$, which directly follows from $M \leq_{\mathtt{ctx}} N$, $\mathtt{weight}([\![M]\!]) = \sum_i p_i$ and $\mathtt{weight}([\![N]\!]) = \sum_j q_j$. □

Theorem 8.40 (Full abstraction). *On $\Lambda_\oplus$-terms:*

1. *relations $\leq_{\mathtt{ctx}}$ and $\lesssim$ coincide;*

2. *relations $=_{\mathtt{ctx}}$ and $\approx$ coincide.*

Proof. $\lesssim$ is complete by Theorem 8.39. The soundness follows from the fact that it is a congruence, which is obtained by exploiting the characterization $\lesssim = \lesssim_{\text{fin}}$.
The result for the equivalences follows from $\approx = \lesssim \cap \lesssim^{-1}$ and $=_{\mathtt{ctx}} = \leq_{\mathtt{ctx}} \cap \leq_{\mathtt{ctx}}^{-1}$. □

8.4 Probabilistic imperative λ-calculus

In this section we add imperative features, namely higher-order references (locations), to the call-by-value calculus, along the lines of the languages in [KW06b; SKS11]. The language is an extension of the one presented in Section 6.1.2. The syntax of terms and values is:

$$
\begin{array}{rll}
M ::= & x & \text{variables} \\
\big| & c & \text{constants} \\
\big| & \lambda x.M & \text{functions} \\
\big| & M_1 M_2 & \text{applications} \\
\big| & l & \text{locations} \\
\big| & (\boldsymbol{\nu}\, x := M_1) M_2 & \text{new location} \\
\big| & !M & \text{dereferencing} \\
\big| & M_1 := M_2 & \text{assignments} \\
\big| & \mathtt{op}(M_1, ..., M_n) & \text{primitive operations} \\
\big| & \mathtt{if}\ M_1\ \mathtt{then}\ M_2\ \mathtt{else}\ M_3 & \text{if-then-else} \\
\big| & \#_i(M) & \text{projection} \\
\big| & (M_1, ..., M_n) & \text{tuples} \\
\big| & M_1 \oplus M_2 & \text{probabilistic choice}
\end{array}
$$

$$V ::= c \;\big|\; \lambda x.M \;\big|\; l \;\big|\; (V_1, ..., V_n)$$

We use s, t to range over stores, i.e., mappings from locations to closed values, and l, k over locations. Then $s[l \to V]$ is the update of s (possibly an extension of s if l is not in the domain of s). The locations that occur in a term M are $\mathtt{Loc}(M)$. We assume that the set of primitive operations contains the equality function on constants, and write $\star$ for the unit value (i.e., the nullary tuple).

The language is typed — a simply-typed system with recursive types — to make sure that the values in the summands of a formal sum have the same structure (e.g., they are all abstractions). We allow recursive types to maintain the peculiar possibility of probabilistic languages of having infinite but meaningful computation trees. Whenever possible, we omit any mention of the types. For instance, in any store update $s[l \to V]$ it is intended that V has the type appropriate for l; in this case we say that the type of V is *consistent* with that of l. In examples, $M_1 \,\underline{\mathtt{seq}}\, M_2$ denotes term $(\lambda.M_2)M_1$, i.e., the execution of M_1 and M_2 in sequence.

Reduction is defined on terms with a store, i.e., configurations of the form $\langle s\,;\, M\rangle$; hence such configurations appear also in formal sums (where we omit brackets). The small-step reduction and the evaluation contexts are defined in Figure 8.3, where we assume that the semantics of primitive operations is already given by the function $\mathtt{Prim}$. The rules for the semantic mapping, $[\![\ ]\!]$, and the multistep reductions relations, $\Longrightarrow$ and $\Longmapsto$, remain those of Figure 8.1, with the addition of a store. In all semantic rules, any configuration $\langle s\,;\, M\rangle$ is *well-formed*, in that M is closed and all the locations in M and s are in the domain $\mathsf{dom}(s)$ of s. As in the previous calculi, it is easy to check that the semantics of a term exists and is unique.

Notations and terminology for (environment) formal sums are adapted to the extended syntax in the expected manner. We only recall the multiplication of an environment formal sum $\mathbf{Y} \overset{\mathtt{def}}{=} \sum_i p_i; s_i; \widetilde{V_i}$ and formal sums $Y_i \overset{\mathtt{def}}{=} \sum_{j\in J_i} p_{i,j}; s_{i,j}; V_{i,j}$ which is defined as:

$$\textstyle\sum_i p_i; \widetilde{V_i} \cdot Y_i \overset{\mathtt{def}}{=} \sum_{i,j\in J_i} p_i \cdot p_{i,j}; s_{i,j}; \widetilde{V_i}, V_{i,j}\ .$$

$$\textsc{Beta}\ \frac{}{\langle s\,;\,(\lambda x.M)V\rangle \longrightarrow 1; s; M\{{}^{V}\!/x\}} \qquad \textsc{Sum}\ \frac{}{\langle s\,;\,M_1 \oplus M_2\rangle \longrightarrow \frac{1}{2}; s; M_1 + \frac{1}{2}; s; M_2}$$

$$\textsc{Assign}\ \frac{}{\langle s\,;\,l := V\rangle \longrightarrow 1; s[l \to V]; \star} \qquad \textsc{Deref}\ \frac{s(l) = V}{\langle s\,;\,!l\rangle \longrightarrow 1; s; V}$$

$$\textsc{New}\ \frac{l \text{ not in the domain of } s}{\langle s\,;\,(\boldsymbol{\nu}\, x := V)M\rangle \longrightarrow 1; s[l \to V]; M\{{}^{l}\!/x\}}$$

$$\textsc{IfTrue}\ \frac{}{\langle s\,;\,\texttt{if true then } M_1 \texttt{ else } M_2\rangle \longrightarrow 1; s; M_1}$$

$$\textsc{IfFalse}\ \frac{}{\langle s\,;\,\texttt{if false then } M_1 \texttt{ else } M_2\rangle \longrightarrow 1; s; M_2}$$

$$\textsc{Proj}\ \frac{}{\langle s\,;\,\#_i(\widetilde{V})\rangle \longrightarrow 1; s; (\widetilde{V})_i} \qquad \textsc{PrimOp}\ \frac{\texttt{Prim}(\texttt{op}, \widetilde{c}) = c'}{\langle s\,;\,\texttt{op}(\widetilde{c})\rangle \longrightarrow 1; s; c'}$$

$$\textsc{Eval}\ \frac{\langle s\,;\,M\rangle \longrightarrow \sum_i p_i; s_i; M_i \quad C \text{ is an evaluation context}}{\langle s\,;\,C[M]\rangle \longrightarrow \sum_i p_i; s_i; C[M_i]}$$

$$\begin{aligned} \text{Evaluation contexts} \quad C := {} & [\cdot] \;\big|\; CM \;\big|\; VC \;\big|\; !C \;\big|\; C := M \;\big|\; l := C \\ & \big|\; \texttt{if } C \texttt{ then } M_1 \texttt{ else } M_2 \;\big|\; (\widetilde{V}, C, \widetilde{M}) \;\big|\; \#_i C \\ & \big|\; \texttt{op}(\widetilde{c}, C, \widetilde{M}) \end{aligned}$$

Figure 8.3: Single-step reduction relation for imperative probabilistic λ-calculus

The context closure of an environment, $(\{M, \widetilde{V_i}\}_i, \{N, \widetilde{W_j}\}_j)^{\widehat{\star}}$, is defined as in the previous section, but now contexts are *location-free*, i.e., no locations occur in the contexts. This constraint, standard in environmental bisimulations for imperative languages, ensures well-formedness of the terms and is not really a limitation because locations may occur in terms of the environments and may thus end up in the terms of the context closure.

8.4.1 Environmental bisimulation

The notion of environmental relation is modified to accommodate stores, which are needed to run terms. The elements of an environmental relation are now well-formed pairs of configurations $(\langle s\,;\,M\rangle, \langle t\,;\,N\rangle)$ or well-formed triples

$$(\mathcal{E}, \textstyle\sum_i p_i; s_i; \widetilde{V_i}, \sum_j q_j; t_j; \widetilde{W_j})\ .$$

Well-formedness on triples ensures that the store s_i of the possible world i defines all locations that appear in s_i, $\widetilde{V_i}$, and $\mathcal{E}_1$, and similarly for t_j, $\widetilde{W_j}$ and $\mathcal{E}_2$. Further, the triples must be *compatible*: the related environment formal sums should have the same length, and should respect the types, that is, corresponding columns of the environment formal sums should contain terms that have the same type.

Since locations could occur in the terms we want to prove equivalent, we parametrize bisimulations with respect to a set $\{\widetilde{l}\}$ of locations such that the pairs of terms in the relation must have stores with domain $\{\widetilde{l}\}$. This allows us to put these locations in the dynamic environment of the relation (clause (1)), which reflects the fact that the locations occurring in the terms are public (i.e., contexts can access them). In what follows, when we write $\{\widetilde{l}\}$ we assume that no repetitions of the same location occur in the sequence $\widetilde{l}$. For a pair $(\{s_i\}_i, \{t_j\}_j)$ of (tuples of) stores, we say that locations $(\{l_i\}_i, \{k_j\}_j)$ are $(\{s_i\}_i, \{t_j\}_j)$*-fresh* if for every i, j we have $l_i \notin \mathsf{dom}(s_i)$ and $k_j \notin \mathsf{dom}(t_j)$.

Definition 8.41 (Environmental bisimulation, imperative)**.** A PE relation is a *(PE)* $\{\widetilde{l}\}$*-bisimulation* if

1. $\langle s\,;\,M\rangle\,\mathcal{R}\,\langle t\,;\,N\rangle$ implies $\mathsf{dom}(s) = \mathsf{dom}(t) = \{\widetilde{l}\}$ and $1; s; \widetilde{l}\,\mathcal{R}_{(M,N)}\,1; t; \widetilde{l}$;

2. $\sum_i p_i; s_i; \widetilde{V_i}\,\mathcal{R}_\mathcal{E}\,\sum_j q_j; t_j; \widetilde{W_j}$ implies:

   (a) $\sum_i p_i = \sum_j q_j$;

   (b) for all r, if $(\widetilde{V_i})_r = \lambda x.M_i$ and $(\widetilde{W_j})_r = \lambda x.N_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\widehat{\star}}$,

   $$\textstyle\sum_i p_i; \widetilde{V_i} \cdot [\![\langle s_i\,;\,M_i\{{}^{T_i}\!/\!_x\}\rangle]\!]\,\mathtt{lift}(\mathcal{R}_\mathcal{E})\,\sum_j q_j; \widetilde{W_j} \cdot [\![\langle t_j\,;\,N_j\{{}^{U_j}\!/\!_x\}\rangle]\!] \;;$$

   (c) for all r, if $(\widetilde{V_i})_r = l_i$ and $(\widetilde{W_j})_r = k_j$ then

   - $\sum_i p_i; s_i; \widetilde{V_i}, s_i(l_i)\,\mathtt{lift}(\mathcal{R}_\mathcal{E})\,\sum_j q_j; t_j; \widetilde{W_j}, t_j(k_j)$,
   - for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\widehat{\star}}$,

   $$\textstyle\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i}\,\mathcal{R}_\mathcal{E}\,\sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j} \;;$$

   (d) for any $(\{s_i\}_i, \{t_j\}_j)$-fresh locations $(\{l_i\}_i, \{k_j\}_j)$, and for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\widehat{\star}}$,

   $$\textstyle\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i}, l_i\,\mathcal{R}_\mathcal{E}\,\sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}, k_j \;;$$

   (e) for all r, if $(\widetilde{V_i})_r = c_i$ and $(\widetilde{W_j})_r = c_j$ then all constants in the two columns are the same (i.e., there is a c with $c_i = c_j = c$ for all i, j);

   (f) for all r, if $(\widetilde{V_i})_r = (V_{i,1}, ..., V_{i,n})$ and $(\widetilde{W_j})_r = (W_{j,1}, ..., W_{j,n})$ then

   $$\textstyle\sum_i p_i; s_i; \widetilde{V_i}, V_{i,1}, ..., V_{i,n}\,\mathtt{lift}(\mathcal{R}_\mathcal{E})\,\sum_j q_j; t_j; \widetilde{W_j}, W_{j,1}, ..., W_{j,n} \;;$$

   (g) $\sum_i p_i; \widetilde{V_i} \cdot [\![\langle s_i\,;\,\mathcal{E}_1\rangle]\!]\,\mathtt{lift}(\mathcal{R}_\mathcal{E})\,\sum_j q_j; \widetilde{W_j} \cdot [\![\langle t_j\,;\,\mathcal{E}_2\rangle]\!]$.

We let $\approx^{\{\widetilde{l}\}}$ denote the union of all $\{\widetilde{l}\}$-bisimulations.

With respect to the definition for pure call-by-value, the definition above has the additional ingredient of the store, and of clauses (2c) and (2d) to deal with the case in which the values are locations: (2c) gives an observer the possibility of reading and writing the store, and (2d) the possibility of extending the store with fresh locations. Clause (2f) adds all elements of a tuple to the dynamic environment. These aspects are similar to those in ordinary environmental bisimulations for imperative languages [SKS11; KLS11].

Three further aspects, however, are new. First, by clause (2e), related environment formal sums should be *first-order consistent*, meaning that corresponding columns of constants should contain exactly one constant. This constraint is a consequence of the equality test on constants in the language. To ensure that first-order consistency is maintained in the bisimulation game, most of the clauses use a lifting construction. Thus, when the evaluation of first-order terms may probabilistically yield different constants, lifting allows us to separate the final possible worlds according to the specific constants obtained. This constraint is further discussed in Examples 8.45 and 8.46. A second new aspect is that, since the effect of the evaluation of the terms in the static environment may change depending on the current store, clauses (1) and (2g) allow us to derive a congruence result for arbitrary terms (not necessarily values), as illustrated in the example below. Finally, we parametrize the relation with a set of locations in order to deal with terms where (public) locations may occur.

In what follows, we sometimes omit any reference to the set of locations parametrizing the relation, and simply refer to (bi)simulations and (bi)similarity when the parametrizing set is not relevant.

Example 8.42. Let $M \overset{\mathsf{def}}{=} l := 1$ and $N \overset{\mathsf{def}}{=}$ $\texttt{if}$ $!l = 0$ $\texttt{then}$ $l := 1$ $\texttt{else}$ Ω. Without the static environment, terms $\langle l = 0\,;\, M\rangle$ and $\langle l = 0\,;\, N\rangle$ are bisimilar. However, they are not contextually equivalent: if $C \overset{\mathsf{def}}{=} [\cdot]\,\underline{\texttt{seq}}\,[\cdot]$, then $\langle l = 0\,;\, C[M]\rangle$ terminates whereas $\langle l = 0\,;\, C[N]\rangle$ does not. This aspect is determined by the store, probabilities do not really matter. Ordinary environmental bisimulations do not have a static environment, and cannot therefore test repeated runs of given terms that are not values; as a consequence M and N are equated, and bisimulation is not fully substitutive on arbitrary terms (see [SKS11, Section 5.2]).

Clause (1) is also modified with respect to pure call-by-name and call-by-value calculi. Indeed, if we defined the clause as follows:

$$\langle s\,;\, M\rangle\,\mathcal{R}\,\langle t\,;\, N\rangle \text{ implies } [\![\langle s\,;\, M\rangle]\!]\,\texttt{lift}(\mathcal{R}_{(M,N)})\,[\![\langle t\,;\, N\rangle]\!]$$

then the bisimulation would not be sound with respect to terms that diverge at the first run. For instance, the terms $M \overset{\mathsf{def}}{=}$ $\texttt{if}$ $!l = 1$ $\texttt{then true else}$ Ω and $N \overset{\mathsf{def}}{=}$ $\texttt{if}$ $!l = 1$ $\texttt{then false else}$ Ω (that are not contextually equivalent thanks to context $C \overset{\mathsf{def}}{=} l := 1\,\underline{\texttt{seq}}\,[\cdot]$) would be bisimilar with store $l = 0$, by simply considering the relation $\{(\langle l = 0\,;\, M\rangle,), \langle l = 0\,;\, N\rangle\}$.

Even if we make the location l public by starting the bisimulation game from terms (M, l) and (N, l) and by exploiting clause (2f), so as to allow contexts to use the location as in [SKS11, Theorem 5.10], we still have that $\langle l = 0\,;\, (M, l)\rangle$ and $\langle l = 0\,;\, (N, l)\rangle$ are bisimilar, since $[\![\langle l = 0\,;\, (M, l)\rangle]\!] = [\![\langle l = 0\,;\, (N, l)\rangle]\!] = \emptyset$.

Hence, clause (1) ensures that location l is actually put in the dynamic environment, so that we can use clause (2c) to change the value of l and then evaluate again M and N using clause (2g).

If the domains of the stores are empty then we can consider bisimulations parametrized by the empty set of locations, and in clause (1) we will have empty sequences of values in the dynamic environment. Anyway, clause (2g) can be applied independently of the presence of values in the dynamic environment.

The following examples are meant to further illustrate and motivate the form and the clauses of our bisimulation. The examples only use boolean and integer locations, and we accordingly assume that all locations in the language are of these types. Higher-order locations would not affect the essence of the examples and would complicate the description of the required bisimulations due to the possibility of extending the store (clause (2d)). (The full abstraction results will not rely on the existence of locations of specific types.) Moreover, since the terms compared always have the same locations, we assume that fresh locations for the extensions of the store are the same on both sides.

Example 8.43 shows that in imperative call-by-value, in contrast with pure call-by-value, to achieve full abstraction it is necessary to define bisimulation on formal sums rather than on terms.

Example 8.43. We have explained in Section 8.1 why the terms

$$H \overset{\mathtt{def}}{=} (\boldsymbol{\nu}\, x := 0)(\lambda.(M \oplus N)) \qquad K \overset{\mathtt{def}}{=} (\boldsymbol{\nu}\, x := 0)(\lambda.M \oplus \lambda.N)$$

where

$$M \overset{\mathtt{def}}{=} \mathtt{if}\ !x = 0\ \mathtt{then}\ x := 1\, \underline{\mathtt{seq}}\, \mathtt{true}\ \mathtt{else}\ \Omega$$
$$N \overset{\mathtt{def}}{=} \mathtt{if}\ !x = 0\ \mathtt{then}\ x := 1\, \underline{\mathtt{seq}}\, \mathtt{false}\ \mathtt{else}\ \Omega$$

are contextually equivalent, but would be separated by a bisimulation that acted on terms. With our bisimulation, we can prove H and K equal using a relation that contains the pair $(\langle s\,;\, H\rangle, \langle s\,;\, K\rangle)$, for s the empty store, and all triples $((H,K), \mathbf{Y}, \mathbf{Z})$ in which $\mathbf{Y}, \mathbf{Z}$ are first-order consistent, have the same total weight and, seeing them as matrices, for every column r of the dynamic environments that is not made of constants one of the following properties holds:

(a) there is l such that all terms in $\mathbf{Y}\lfloor_r$ are $\lambda.(M \oplus N)\{{}^{l}\!/\!{}_{x}\}$, whereas all terms in $\mathbf{Z}\lfloor_r$ are either $\lambda.M\{{}^{l}\!/\!{}_{x}\}$ or $\lambda.N\{{}^{l}\!/\!{}_{x}\}$; moreover l does not occur elsewhere in terms of the dynamic environment and its value in the stores is 1;

(b) there is l such that all terms in $\mathbf{Y}\lfloor_r$ are $\lambda.(M \oplus N)\{{}^{l}\!/\!{}_{x}\}$, whereas $\mathbf{Z}\lfloor_r$ contains both $\lambda.M\{{}^{l}\!/\!{}_{x}\}$ and $\lambda.N\{{}^{l}\!/\!{}_{x}\}$; moreover l does not occur elsewhere in the dynamic environment and its value in the stores is 0. The right-hand matrix obtained by erasing all columns that are not of this shape is either $\emptyset$ or (without considering the stores) of the form $(...((Y_1 \cdot Y_2) \cdot Y_3) \cdot ...) \cdot Y_n$, for $Y_i = \frac{1}{2}; \lambda.M\{{}^{l_i}\!/\!{}_{x}\} + \frac{1}{2}; \lambda.N\{{}^{l_i}\!/\!{}_{x}\}$. This clause guarantees that $\lambda.M\{{}^{l_i}\!/\!{}_{x}\}$ and $\lambda.N\{{}^{l_i}\!/\!{}_{x}\}$ have the same probability in every column (if l_i is set to 0 in the stores), and that this property still holds if the matrix is splitted by separating the rows with $\lambda.M\{{}^{l_i}\!/\!{}_{x}\}$ from the rows with $\lambda.N\{{}^{l_i}\!/\!{}_{x}\}$;

(c) there is l such that all terms in $\mathbf{Y}\lfloor_r$ and $\mathbf{Z}\lfloor_r$ are l; moreover l is set to the same value in all the stores.

Finally, we add to the relation the triple $((H,K),1;s;\emptyset,1;s;\emptyset)$, where $\emptyset$ denotes the empty dynamic environment, to satisfy clause (1) of Definition 8.41. Clause (2g) is handled appealing to item (b). The most interesting case is the bisimulation clause (2b) applied to a column r of functions that satisfy item (b). The result of the evaluation of such functions (with $\star$ as argument) is that l is set to 1 and then `true` and `false` are returned, with the same probability. Using the lifting construction we can now split the possible worlds in which `true` has been produced and those in which `false` has been produced, yielding two pairs of environment formal sums both of which are in the bisimulation (note that the lifting splits the original column r so that the corresponding column in the two final pairs satisfies item (a) above).

In this work we sometimes view environment formal sums as matrices (Figure 8.2). This however is only for representation convenience: our environments are *tuples of rows* (each row representing a possible world originated by the probabilistic evaluation of terms), rather than *tuples of columns*, that is, tuples of formal sums. The next example shows that if the environments were tuples of formal sums, where formal sums are added to the environment following the evaluation of terms during the bisimulation game, then bisimilarity would not be complete. Intuitively this happens because the histories of different possible worlds would not be anymore separated and could interfere.

Example 8.44. Let

$$A \stackrel{\mathtt{def}}{=} (\boldsymbol{\nu}\, y := 0)(L \oplus M) \quad B \stackrel{\mathtt{def}}{=} (\boldsymbol{\nu}\, y := 0)(L \oplus N)$$
$$L \stackrel{\mathtt{def}}{=} \lambda.!y \quad M \stackrel{\mathtt{def}}{=} \lambda.(y := 1 \,\underline{\mathtt{seq}}\, 2) \quad N \stackrel{\mathtt{def}}{=} \lambda.2\ .$$

Terms A and B create a new location and allow the reading capability on it in the subterm L. The writing capability, in contrast, exists only in the subterm M of A. A behavior from A that could not be mimicked with B is the run of M, where 1 is assigned to the location x, followed by a run of L, where x is read and 1 is emitted (with B, any value produced by L would be 0). This behavior, however, is impossible, because L and M are in a probabilistic choice and are therefore obtained in two distinct possible worlds, in one of which x can only be read, in the other x can only be written. Moreover, the writing capability alone is irrelevant, because the location is private; hence it can be omitted from M, resulting in the term N that appears in B. Indeed, A and B are contextually equivalent.

However, the ‘wrong’ behavior above for A could be reproduced in the bisimulation if the environments were tuples of formal sums (that is, all possible worlds have the same environment, made of formal sums). The formal sum obtained by the evaluation of A, with summand terms L and M, would be stored in the environment and could then be executed several times, with possible interleaving of evaluations of L and M. (The example could be made more complex so as to obtain a ‘wrong’ behavior from the execution of two different formal sums in the environment, rather than by multiple executions of the same formal sum.)

With our bisimulation, we can prove A, B equal using a relation composed by (A,B) (for simplicity, we leave out the store) and by all triples $((A,B),\mathbf{Y},\mathbf{Z})$ where the environment formal sums $\mathbf{Y} = 1;s;V_1,...,V_n$ and $\mathbf{Z} = 1;t;W_1,...,W_n$ are first-order consistent, and for each column r that does not contain constants one of the following holds:

(a) there is l such that $V_r = L\{{}^{l}\!/\!{}_{y}\} = W_r$; moreover l does not occur elsewhere in the dynamic environment or within a location of the stores, and is set to 0 in both stores;

(b) there is l such that $V_r = M\{{}^{l}\!/\!{}_{y}\}$ and $W_r = N$; and, again, l does not occur elsewhere in the dynamic environment or within a location of the stores; moreover in the store s we have $s(l) \in \{0, 1\}$ whereas in t we have $t(l) = 0$;

(c) $V_r = W_r = l$ for some l assigned to the same value in both stores.

The proof that this relation is a bisimulation crucially exploits the lifting construction. For instance, using (a) and (b) one shows that the semantics of A and B are in the lifting of the relation, and similarly one proceeds when handling clause (2g) of the bisimulation.

The main purpose of the lifting construct in Definition 8.41 of environmental bisimulation is to maintain the first-order consistency of related environment formal sums. One may wonder whether something simpler would suffice, namely avoiding the lifting construct altogether and simply requiring that, whenever two first-order terms are evaluated, the probability of obtaining a given constant is the same on both sides (and thus maintaining fist-order consistency by avoiding the addition of such values onto the dynamic environments). The example below shows that this would be unsound.

Example 8.45. We compare the terms $A \overset{\text{def}}{=} (\boldsymbol{\nu}\, x{:=}0)(M, N_1)$ and $B \overset{\text{def}}{=} (\boldsymbol{\nu}\, x{:=}0)(M, N_2)$ where

$$M \overset{\text{def}}{=} \lambda.\ \texttt{if}\ !x = 0\ \texttt{then}\ ((x := 1\, \underline{\texttt{seq}}\, \texttt{true}) \oplus (x := 2\, \underline{\texttt{seq}}\, \texttt{false}))\ \texttt{else}\ \Omega$$
$$N_1 \overset{\text{def}}{=} \lambda.\ \texttt{if}\ !x = 2\ \texttt{then}\ x := 3\, \underline{\texttt{seq}}\, n\ \texttt{else}\ \Omega$$
$$N_2 \overset{\text{def}}{=} \lambda.\ \texttt{if}\ (!x = 1 \vee !x = 2)\ \texttt{then}\ x := 3\, \underline{\texttt{seq}}\, (n \oplus \Omega)\ \texttt{else}\ \Omega$$

and n is any integer. The terms A and B produce the values $(M, N_1)\{{}^{l}\!/\!{}_{x}\}$ and $(M, N_2)\{{}^{l}\!/\!{}_{x}\}$ and l is a location that is accessible only to such values. The definitions of $M\{{}^{l}\!/\!{}_{x}\}$ and $N_i\{{}^{l}\!/\!{}_{x}\}$ (for $i = 1, 2$) use conditionals on the content of l in such a way that the only meaningful manipulations with the values $(M\{{}^{l}\!/\!{}_{x}\}, N_i\{{}^{l}\!/\!{}_{x}\})$ is to evaluate $M\{{}^{l}\!/\!{}_{x}\}$ first, and then, possibly, to evaluate $N_i\{{}^{l}\!/\!{}_{x}\}$. Any other order of evaluation would produce a divergence.

We explain why, intuitively, bisimilarity would equate A and B if, on constants, bisimulation simply checked the probabilities of obtaining each constant (rather than employing the lifting construction). The evaluation of (the body of) $M\{{}^{l}\!/\!{}_{x}\}$ produces `true` or `false`, with the same probability $\frac{1}{2}$ and with l respectively set to 1 and 2. Then the only meaningful observation is the evaluation of the values $N_i\{{}^{l}\!/\!{}_{x}\}$. This means evaluating the formal sums

$$F_1 \overset{\text{def}}{=} \tfrac{1}{2}; l = 1; N_1\{{}^{l}\!/\!{}_{x}\}\star + \tfrac{1}{2}; l = 2; N_1\{{}^{l}\!/\!{}_{x}\}\star$$
$$\text{and}\ \ F_2 \overset{\text{def}}{=} \tfrac{1}{2}; l = 1; N_2\{{}^{l}\!/\!{}_{x}\}\star + \tfrac{1}{2}; l = 2; N_2\{{}^{l}\!/\!{}_{x}\}\star\ .$$

The evaluation of F_1 terminates only when $l = 2$, yielding the value formal sum $Y_1 \overset{\text{def}}{=} \frac{1}{2}; l = 3; n$. The evaluation of F_2, in contrast, may terminate under both stores, yielding the value formal sum $Y_2 \overset{\text{def}}{=} \frac{1}{4}; l = 3; n + \frac{1}{4}; l = 3; n$. Both in Y_1 and in Y_2 the outcome n has the overall probability $\frac{1}{2}$.

The terms A and B however are not contextually equivalent, because distinguished by a context C that evaluates $M\{{}^{l}\!/\!{}_{x}\}$ and then proceeds with the evaluation $N_i\{{}^{l}\!/\!{}_{x}\}$ *only* when the outcome from $M\{{}^{l}\!/\!_{x}\}$ was `true`. Now, $C[A]$ never terminates, whereas $C[B]$ terminates and produces n with probability $\frac{1}{4}$.

Our environmental bisimulation distinguishes A from B because we separately analyze the possible worlds in which the evaluation of $M\{{}^{l}\!/\!_{x}\}$ has produced `true` and the possible worlds in which the evaluation has produced `false`, somehow mimicking the effect of the context C above.

Yet another possibility for avoiding the lifting construct of the Definition 8.41 of bisimulation might have been to drop the requirement of first-order consistency, allowing environment formal sums in which a first-order column may contain different constants. Thus constants would be added to the dynamic environment as any other type of value, and one would simply check that, at any time, the weights for the occurrences of a given constant in related columns are the same; formally, replacing clause (2e) with:

(2e′) for every column r and every constant c,
$\sum_{\{i \mid (\widetilde{V_i})_r = c\}} p_i = \sum_{\{j \mid (\widetilde{W_j})_r = c\}} q_j$.

Example 8.46 shows that this choice would be unsound too. We write $(\boldsymbol{\nu}\, x,\, y := 0)M$ for the creation of two locations in which the initialization of the first one is irrelevant.

Example 8.46. This is a variation of the previous example.

We compare the terms $M_1 \overset{\text{def}}{=} (\boldsymbol{\nu}\, x,\, y{:=}0)(A, B_1)$ and $M_2 \overset{\text{def}}{=} (\boldsymbol{\nu}\, x,\, y{:=}0)(A, B_2)$ where

$$\begin{aligned}
A &\overset{\text{def}}{=} \lambda.\ \texttt{if}\ !y = 0\ \texttt{then}\ y := 1\ \underline{\texttt{seq}}\ (\lambda z.(x := z\ \underline{\texttt{seq}}\ z))(\texttt{true} \oplus \texttt{false})\ \texttt{else}\ \Omega \\
B_1 &\overset{\text{def}}{=} \lambda.\ \texttt{if}\ !y = 1\ \texttt{then}\ y := 2\ \underline{\texttt{seq}}\ !x\ \texttt{else}\ \Omega \\
B_2 &\overset{\text{def}}{=} \lambda.\ \texttt{if}\ !y = 1\ \texttt{then}\ y := 2\ \underline{\texttt{seq}}\ (\texttt{true} \oplus \texttt{false})\ \texttt{else}\ \Omega
\end{aligned}$$

As in the previous example, M_1 and M_2 respectively yield the values $(A, B_1)\{{}^{l,l'}\!/\!_{x,y}\}$ and $(A, B_2)\{{}^{l,l'}\!/\!_{x,y}\}$, and the interactions of the terms with the store is such that the only meaningful experiment is to evaluate $A\{{}^{l,l'}\!/\!_{x,y}\}$ first, and then $B_i\{{}^{l,l'}\!/\!_{x,y}\}$ (indeed, the location l' is only used to this end).

We explain why, intuitively, the variant (2e′) above of the clause for first-order values would incorrectly equate M_1 and M_2. The evaluation of $A\{{}^{l,l'}\!/\!_{x,y}\}$ adds to the dynamic environment the formal sum

$$\tfrac{1}{2}; l = \texttt{true}, l' = 1; \texttt{true} + \tfrac{1}{2}; l = \texttt{false}, l' = 1; \texttt{false}$$

(the values produced are also placed in the location l).

Now, in one case the evaluation of $B_1\{{}^{l,l'}\!/\!_{x,y}\}$ adds to the dynamic environment a column of boolean values identical to the column produced above (because $B_1\{{}^{l,l'}\!/\!_{x,y}\}$ emits the value stored in l, which is identical to the value produced by the evaluation of $A\{{}^{l,l'}\!/\!_{x,y}\}$). This means that we end up with an environment formal sum in which the relevant columns are

$$\begin{array}{lll}
\frac{1}{2}\ ; & \texttt{true}\ , & \texttt{true} \\
\frac{1}{2}\ ; & \texttt{false}\ , & \texttt{false}
\end{array} \tag{8.5}$$

In contrast, when evaluating $B_2\{{}^{l,l'}\!/\!_{x,y}\}$ each possible world is split into two, in each of which `true` and `false` have probability $\frac{1}{2}$. Thus the relevant columns of the final environment formal sum are

$$\begin{array}{ll}
\frac{1}{4}\,;\ \texttt{true}\,, & \texttt{true} \\
\frac{1}{4}\,;\ \texttt{true}\,, & \texttt{false} \\
\frac{1}{4}\,;\ \texttt{false}\,, & \texttt{true} \\
\frac{1}{4}\,;\ \texttt{false}\,, & \texttt{false}
\end{array}$$

In each of these columns, the probabilities for `true` and `false` are the same as in the columns of (8.5), as required by (2e$'$).

However the terms are not contextual equivalent. They are separated by a context that evaluates $B_i\{{}^{l,\,l'}\!/\!_{x,y}\}$ only if the outcome of the evaluation of $A\{{}^{l,\,l'}\!/\!_{x,y}\}$ is `true`. Thus the overall probability of obtaining `true` at the end is $\frac{1}{2}$ in one case, and $\frac{1}{4}$ in the other. Similarly the terms are distinguished in our bisimulation, reasoning along the lines of Example 8.45.

The definition of $\{\widetilde{l}\}$-simulation is the same as the definition of $\{\widetilde{l}\}$-bisimulation, but for the first clause on environment formal sums with stores, which becomes: $\sum_i p_i \leq \sum_j q_j$. We let $\lesssim^{\{\widetilde{l}\}}$ denote $\{\widetilde{l}\}$-similarity.

The basic properties and definitions for environmental (bi)simulations in pure call-by-value remain valid, with the due adjustments. In some cases, however, some subtleties arise.

It can be easily proved that, for any set $\{\widetilde{l}\}$, $\{\widetilde{l}\}$-bisimilarity and $\{\widetilde{l}\}$-similarity are an equivalence and a preorder relation respectively. For proving transitivity, in particular, the restriction to parametrized relations, rather than to arbitrary relations, is fundamental. Analogously, we have that $\{\widetilde{l}\}$-(bi)simulations are closed under union, and thus relations $\approx^{\{\widetilde{l}\}}$ and $\lesssim^{\{\widetilde{l}\}}$ are respectively the largest $\{\widetilde{l}\}$-bisimulation and $\{\widetilde{l}\}$-simulation.

In finite-step simulation, clauses (2b) and (2g) are modified so to make sure that only a finite number of reductions are performed on the challenger side. No modification is made to the clauses (1), (2c), (2d), (2e) and (2f) for locations, constants, and tuples, because there is no evaluation of terms involved.

The definition of extended environment formal sum and of the multi-step reduction from extended environment formal sums to environment formal sums is adapted to the imperative case as expected, by assuming that when $\sum_i p_i; s_i; \widetilde{V}_i; M_i \Longmapsto Y$ there is only a finite number of $\langle s_i\,;\, M_i\rangle$ that actually perform some reduction steps.

Definition 8.47. A PE relation is a *finite-step* $\{\widetilde{l}\}$*-simulation* if it satisfies the same clauses (1), (2a), (2c), (2d), (2e) and (2f) of (the simulation version of) Definition 8.41; and, in place of clauses (2b) and (2g) we have:

(2) $\sum_i p_i; s_i; \widetilde{V}_i \;\mathcal{R}_{\mathcal{E}}\; \sum_j q_j; t_j; \widetilde{W}_j$ implies:

- (b) for all r, if $(\widetilde{V}_i)_r = \lambda x.M_i$ and $(\widetilde{W}_j)_r = \lambda x.N_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V}_i\}_i, \{\mathcal{E}_2, \widetilde{W}_j\}_j)^\star$, if $\sum_i p_i; s_i; \widetilde{V}_i; M_i\{{}^{T_i}\!/\!_x\} \Longmapsto Y$ then $Y\,\texttt{lift}(\mathcal{R}_{\mathcal{E}})\,\sum_j q_j; \widetilde{W}_j \cdot [\![\langle t_j\,;\, N_j\{{}^{U_j}\!/\!_x\}\rangle]\!]$;
- (g) if $\sum_i p_i; s_i; \widetilde{V}_i; \mathcal{E}_1 \Longmapsto Y$ then $Y\,\texttt{lift}(\mathcal{R}_{\mathcal{E}})\,\sum_j q_j; \widetilde{W}_j \cdot [\![\langle t_j\,;\, \mathcal{E}_2\rangle]\!]$.

We write $\lesssim^{\{\widetilde{l}\}}_{\mathrm{fin}}$ for finite-step $\{\widetilde{l}\}$-similarity. We prove that $\{\widetilde{l}\}$-similarity and finite-step $\{\widetilde{l}\}$-similarity coincide by exploiting the saturation by approximants and the saturation by suprema of $\{\widetilde{l}\}$-simulations and finite-step $\{\widetilde{l}\}$-simulations, respectively. Since only clauses (2b) and (2g) are modified, we can proceed as in the proofs for the pure calculi.

Theorem 8.48. $\lesssim^{\{\widetilde{l}\}} = \lesssim^{\{\widetilde{l}\}}_{\text{fin}}$.

Precongruence is derived for the finite-step similarity using 'up-to lifting and environment' techniques, and then transported to similarity, from which it is transported to bisimilarity using the characterization of bisimilarity as the equivalence induced by the simulation preorder.

The up-to lifting and environment technique is defined analogously to the probabilistic call-by-value case. The environment preorder is as follows: $(\mathbf{Y},\mathbf{Z}) \leq_{\texttt{env}} (\mathbf{Y}',\mathbf{Z}')$ if $\mathbf{Y} = \sum_i p_i; s_i; \widetilde{V_i}, \mathbf{Z} = \sum_j q_j; t_j; \widetilde{W_j}$ with $\mid \mathbf{Y} \mid = \mid \mathbf{Z} \mid$ and $\mathbf{Y}' = \sum_i p_i; s_i; \widetilde{V_i'}, \mathbf{Z}' = \sum_j q_j; t_j; \widetilde{W_j'}$ with $\mid \mathbf{Y}' \mid = \mid \mathbf{Z}' \mid$ and for every index r in $\mid \mathbf{Y} \mid$ there is an index r' in $\mid \mathbf{Y}' \mid$ such that for all i, j, $(\widetilde{V_i})_r = (\widetilde{V_i'})_{r'}$ and $(\widetilde{W_j})_r = (\widetilde{W_j'})_{r'}$.

Definition 8.49. A PE relation is a *finite-step $\{\widetilde{l}\}$-simulation up-to lifting and environment* if:

1. $\langle s\,;\, M\rangle \,\mathcal{R}\, \langle t\,;\, N\rangle$ implies $\mathsf{dom}(s) = \mathsf{dom}(t) = \{\widetilde{l}\}$ and $1; s; \widetilde{l}\; \mathcal{R}_{(M,N)}\; 1; t; \widetilde{l}$;

2. $\sum_i p_i; s_i; \widetilde{V_i}\; \mathcal{R}_{\mathcal{E}}\; \sum_j q_j; t_j; \widetilde{W_j}$ implies:

   (a) $\sum_i p_i \leq \sum_i q_i$;

   (b) for all r, if $(\widetilde{V_i})_r = \lambda x.M_i$ and $(\widetilde{W_j})_r = \lambda x.N_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\star}$, if $\sum_i p_i; s_i; \widetilde{V_i}; M_i\{^{T_i}\!/\!_x\} \Longmapsto Y$ then $Y\, \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{\mathcal{E}}))\; \sum_j q_j; \widetilde{W_j} \cdot [\![\langle t_j\,;\, N_j\{^{U_j}\!/\!_x\}\rangle]\!]$;

   (c) for all r, if $(\widetilde{V_i})_r = l_i$ and $(\widetilde{W_j})_r = k_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\star}$,

      - $\sum_i p_i; s_i; \widetilde{V_i}, s_i(l_i)\, \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{\mathcal{E}}))\; \sum_j q_j; t_j; \widetilde{W_j}, t_j(k_j)$,
      - $\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i}\, \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{\mathcal{E}}))\, \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}$;

   (d) for any $(\{s_i\}_i, \{t_j\}_j)$-fresh locations $(\{l_i\}_i, \{k_j\}_j)$, and for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V_i}\}_i, \{\mathcal{E}_2, \widetilde{W_j}\}_j)^{\star}$,

   $$\textstyle\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i}, l_i\, \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{\mathcal{E}}))\; \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}, k_j\, ;$$

   (e) for all r, if $(\widetilde{V_i})_r = c_i$ and $(\widetilde{W_j})_r = c_j$ then all constants in the two columns are the same (i.e., there is c_a with $c_i = c_j = c_a$ for all i, j) ;

   (f) for all r, if $(\widetilde{V_i})_r = (V_{i,1}, ..., V_{i,n})$ and $(\widetilde{W_j})_r = (W_{j,1}, ..., W_{j,n})$ then

   $$\textstyle\sum_i p_i; s_i; \widetilde{V_i}, V_{i,1}, ..., V_{i,n}\, \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{\mathcal{E}}))\; \sum_j q_j; t_j; \widetilde{W_j}, W_{j,1}, ..., W_{j,n}\, ;$$

   (g) $\sum_i p_i; s_i; \widetilde{V_i}; \mathcal{E}_1 \Longmapsto Y$ then $Y\, \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}_{\mathcal{E}}))\; \sum_j q_j; \widetilde{W_j} \cdot [\![\langle t_j\,;\, \mathcal{E}_2\rangle]\!]$.

Theorem 8.50. *If $\mathcal{R}$ is a finite-step $\{\widetilde{l}\}$-simulation up-to lifting and environment then* $\mathcal{R} \subseteq \lesssim^{\{\widetilde{l}\}}_{\text{fin}}$.

The soundness of the up-to lifting and environment technique follows as in call-by-value (Lemma 8.37). Given a finite-step $\{\widetilde{l}\}$-simulation up-to lifting and environment $\mathcal{R}$, we prove that $\mathcal{S}= \mathtt{Pairs}(\mathcal{R}) \cup \bigcup_{\mathcal{E}} \mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}_{\mathcal{E}}))$ is a finite-step $\{\widetilde{l}\}$-simulation.

We first prove congruence of finite-step $\{\widetilde{l}\}$-similarity for contexts with locations in $\{\widetilde{l}\}$, and then we show how to derive congruence for general contexts. The proofs of these results are reported in Section 8.5. The proof structure for Theorem 8.51 is as in call-by-value; we define the context closure of a finite-step $\{\widetilde{l}\}$-simulation and we prove that it is a finite-step $\{\widetilde{l}\}$-simulation up-to lifting and environment.

Theorem 8.51. *Finite-step $\{\widetilde{l}\}$-similarity is a precongruence for contexts with locations in $\{\widetilde{l}\}$: if $\langle s\,;\, M\rangle \lesssim_{\mathrm{fin}}^{\{\widetilde{l}\}} \langle t\,;\, N\rangle$ then $\langle s\,;\, C[M]\rangle \lesssim_{\mathrm{fin}}^{\{\widetilde{l}\}} \langle t\,;\, C[N]\rangle$, for every C with $\mathtt{Loc}(C) \subseteq \{\widetilde{l}\}$.*

Then we derive precongruence for general contexts by showing how to move from relations parametrized by a set $\{\widetilde{l}\}$ to relations parametrized by $\{\widetilde{l'}\}$, for $\{\widetilde{l'}\}$ a set including $\{\widetilde{l}\}$.

Theorem 8.52. *Let $\widetilde{l'} = \widetilde{l}, \widetilde{l''}$ and let $\widetilde{V'} = \widetilde{V}, \widetilde{V''}$ be a sequence of values whose types are consistent with those of $\widetilde{l'}$, and with locations in $\{\widetilde{l'}\}$. Let C be a context with locations in $\{\widetilde{l'}\}$. If $\langle s\,;\, M\rangle \lesssim_{\mathrm{fin}}^{\{\widetilde{l}\}} \langle t\,;\, N\rangle$ then:*

- $\langle s[\widetilde{l''} \to \widetilde{V''}]\,;\, C[M]\rangle \lesssim_{\mathrm{fin}}^{\{\widetilde{l'}\}} \langle t[\widetilde{l''} \to \widetilde{V''}]\,;\, C[N]\rangle$;
- $\langle \widetilde{l'} \to \widetilde{V'}\,;\, C[M]\rangle \lesssim_{\mathrm{fin}}^{\{\widetilde{l'}\}} \langle \widetilde{l'} \to \widetilde{V'}\,;\, C[N]\rangle$.

8.4.2 Contextual equivalence

We set $\langle s\,;\, M\rangle \Downarrow = \mathtt{weight}(\llbracket\langle s\,;\, M\rangle\rrbracket)$. Contextual equivalence and the contextual preorder are defined by quantifying over all stores and contexts.

Definition 8.53. M and N are in the *contextual preorder*, written $M \leq_{\mathtt{ctx}} N$, (resp. *contextually equivalent*, written $M =_{\mathtt{ctx}} N$), if, for any store s and context C such that $\langle s\,;\, C[M]\rangle$ and $\langle s\,;\, C[N]\rangle$ are well-formed, $\langle s\,;\, C[M]\rangle\Downarrow \leq \langle s\,;\, C[N]\rangle\Downarrow$ (resp. $\langle s\,;\, C[M]\rangle\Downarrow = \langle s\,;\, C[N]\rangle\Downarrow$).

Theorem 8.54 (Completeness)**.** *Let $\mathtt{Loc}(M) \cup \mathtt{Loc}(N) \subseteq \{\widetilde{l}\}$. If $M \leq_{\mathtt{ctx}} N$ then $\langle \widetilde{l} = \widetilde{V}\,;\, M\rangle \lesssim^{\{\widetilde{l}\}} \langle \widetilde{l} = \widetilde{V}\,;\, N\rangle$, for any $\widetilde{V}$ whose types and locations are consistent with $\widetilde{l}$.*

Proof. We prove that the relation

$$
\begin{aligned}
\mathcal{R} = \{ & (\langle \widetilde{l} = \widetilde{V}\,;\, M\rangle, \langle \widetilde{l} = \widetilde{V}\,;\, N\rangle) \mid M \leq_{\mathtt{ctx}} N \ \wedge\ \{\widetilde{l}\} = \mathtt{Loc}(M) \cup \mathtt{Loc}(N)\} \cup \\
& \{((M,N), \textstyle\sum_i p_i; s_i; V_1^i, ..., V_n^i\,, \sum_j q_j; t_j; W_1^j, ..., W_n^j) \mid M \leq_{\mathtt{ctx}} N \\
& \wedge \exists C, \widetilde{V} \text{ such that } (\llbracket\langle \widetilde{l} = \widetilde{V}\,;\, C[M]\rangle\rrbracket = \textstyle\sum_i p_i; s_i; \lambda x.x V_1^i ... V_n^i \\
& \wedge \llbracket\langle \widetilde{l} = \widetilde{V}\,;\, C[N]\rangle\rrbracket = \textstyle\sum_j q_j; t_j; \lambda x.x W_1^j ... W_n^j \\
& \text{with } \mathtt{Loc}(C) \cup \mathtt{Loc}(M) \cup \mathtt{Loc}(N) \subseteq \{\widetilde{l}\} \\
& \wedge\ \text{they are } \textit{first-order consistent})\}
\end{aligned}
$$

is a $\{\widetilde{l}\}$-simulation. The full proof is in Section 8.5. □

We can now derive full abstraction for the simulation preorder and bisimilarity. Contextual equivalence and preorder are defined on terms, while bisimilarity and the simulation preorder are defined over configurations of a term and a store. In the full abstraction result we show that congruence on terms corresponds to bisimilarity when an arbitrary store is considered.

Theorem 8.55 (Full abstraction)**.** *Let $\{\widetilde{l}\}$ be the set of locations that occur in M or N. We have, for any $\widetilde{V}$ whose types and locations are consistent with $\widetilde{l}$:*

- *$M \leq_{\mathtt{ctx}} N$ if and only if $\langle \widetilde{l} = \widetilde{V} \,;\, M\rangle \lesssim^{\{\widetilde{l}\}} \langle \widetilde{l} = \widetilde{V} \,;\, N\rangle$;*
- *$M =_{\mathtt{ctx}} N$ if and only if $\langle \widetilde{l} = \widetilde{V} \,;\, M\rangle \approx^{\{\widetilde{l}\}} \langle \widetilde{l} = \widetilde{V} \,;\, N\rangle$.*

Proof. Theorem 8.54 proves completeness. For soundness, suppose that $\langle \widetilde{l} = \widetilde{V} \,;\, M\rangle \lesssim^{\{\widetilde{l}\}} \langle \widetilde{l} = \widetilde{V} \,;\, N\rangle$ for some $\widetilde{V}$ consistent with $\widetilde{l}$. Let s be a store and C a context such that $\langle s \,;\, C[M]\rangle$ and $\langle s \,;\, C[N]\rangle$ are well-formed. We want to prove that $\langle s \,;\, C[M]\rangle \Downarrow \leq \langle s \,;\, C[N]\rangle \Downarrow$.
By well-formedness, we know that $s = \emptyset[\widetilde{l'} \to \widetilde{V'}]$, for some $\widetilde{l'}$ such that $\widetilde{l'} = \widetilde{l}, \widetilde{l''}$ and for some consistent $\widetilde{V'}$, and that C has locations in $\{\widetilde{l'}\}$. By $\langle \widetilde{l} = \widetilde{V} \,;\, M\rangle \lesssim^{\{\widetilde{l}\}} \langle \widetilde{l} = \widetilde{V} \,;\, N\rangle$ we have $\langle \widetilde{l} = \widetilde{V} \,;\, M\rangle \lesssim^{\{\widetilde{l}\}}_{\text{fin}} \langle \widetilde{l} = \widetilde{V} \,;\, N\rangle$ and by Theorem 8.52 we derive $\langle \widetilde{l'} = \widetilde{V'} \,;\, C[M]\rangle \lesssim^{\{\widetilde{l}\}}_{\text{fin}} \langle \widetilde{l'} = \widetilde{V'} \,;\, C[N]\rangle$. Then $\langle \widetilde{l'} = \widetilde{V'} \,;\, C[M]\rangle \Downarrow \leq \langle \widetilde{l'} = \widetilde{V'} \,;\, C[N]\rangle \Downarrow$.

□

The universal quantification over stores in the full abstraction is outside, and not inside, the double implication, i.e., we do not prove

$$M \leq_{\mathtt{ctx}} N \text{ if and only if for all consistent } \widetilde{V},\ \langle \widetilde{l} = \widetilde{V} \,;\, M\rangle \lesssim^{\{\widetilde{l}\}} \langle \widetilde{l} = \widetilde{V} \,;\, N\rangle$$

but rather

$$\text{for all consistent } \widetilde{V},\ M \leq_{\mathtt{ctx}} N \text{ if and only if } \langle \widetilde{l} = \widetilde{V} \,;\, M\rangle \lesssim^{\{\widetilde{l}\}} \langle \widetilde{l} = \widetilde{V} \,;\, N\rangle.$$

The former statement implies the latter one, since the latter allows us to only consider one store. The reason why we can use the latter one, and thereby consider an arbitrary store, is that the definition of simulation and bisimulation already includes the universal quantification over different assignments of the locations in $\widetilde{l}$, since the locations are in the dynamic environment and we can apply the second item of clause (2c), as we have seen in the proof of Theorem 8.55.

8.5 Proofs

Proof of Lemma 8.11

1. The proof of (1) follows from the definition of $[\![M]\!]$ as the supremum of the set $\{Y \mid M \Longmapsto Y\}$ with respect to the $\leq_{\mathsf{apx}}$ preorder. Let $\mathcal{R}$ be a simulation and let $\mathcal{S}$ be its saturation by approximants.
If $M \;\mathcal{S}\; N$ then $[\![M]\!] \;\mathcal{S}_{(M,N)}\; [\![N]\!]$, since $\leq_{\mathsf{apx}}$ is reflexive and $[\![M]\!] \;\mathcal{R}_{(M,N)}\; [\![N]\!]$. If $Y \;\mathcal{S}_{(M,N)}\; Z$ then there is a Y' such that $Y \leq_{\mathsf{apx}} Y'$ and $Y' \;\mathcal{R}_{(M,N)}\; Z$. We have

that $\texttt{weight}(Y) \leq \texttt{weight}(Y')$, by the definition of approximant, and $\texttt{weight}(Y') \leq \texttt{weight}([\![Z]\!])$, since $\mathcal{R}$ is a simulation.
Suppose $Y + Y'' = Y'$. Then, for any context C,

$$[\![Y \bullet C[M]]\!] \leq_{\texttt{apx}} [\![Y \bullet C[M]]\!] + [\![Y'' \bullet C[M]]\!] = [\![Y' \bullet C[M]]\!] \; \mathcal{R}_{(M,N)} \; [\![Z \bullet C[N]]\!]$$

which implies $[\![Y \bullet C[M]]\!] \; \mathcal{S}_{(M,N)} \; [\![Z \bullet C[N]]\!]$.

2. Let $\mathcal{S} = \bigcup_n \mathcal{R}^n$ be the saturation by suprema of a finite-step simulation $\mathcal{R}$. The clause on λ-terms is immediate, since $M \; \mathcal{S} \; N$ and $M \Longmapsto Y$ implies $Y \; \mathcal{R}^0_{(M,N)} \; [\![N]\!]$ (by the definition of finite-step simulation), which in turn implies that $[\![M]\!] \; \mathcal{R}^1_{(M,N)} \; [\![N]\!]$ (since $[\![M]\!] = \sup\{Y \mid M \Longmapsto Y\}$ and thus we can find an ordered sequence of formal sums that satisfies the condition for $\mathcal{R}^1$).
For the clause on formal sums, the crux is proving the following lemma.

Lemma 8.56. *If* $\sum_i p_i; \lambda x.M_i \; \mathcal{R}^n_{\mathcal{E}} \; \sum_j q_j; \lambda x.N_j$ *then:*

(a) $\sum_i p_i \leq \sum_j q_j$
(b) $\sum_i p_i; M_i\{P/x\} \Longmapsto Y$ *implies* $Y \; \mathcal{R}^n_{\mathcal{E}} \; \sum_j q_j \cdot [\![N_j\{Q/x\}]\!]$*, for all* $P, Q \in \mathcal{E}^\star$*.*

Proof. The proof is by induction on n.
For the case $n = 0$, the two properties above are immediate consequences of the definition of $\mathcal{R}$.
For the inductive case, if $Y \, \mathcal{R}^{n+1}_{(M,N)} Z$ then either $Y \, \mathcal{R}^{n}_{(M,N)} Z$, and the result follows by the inductive hypothesis, or $Y = \sup S$ for $S = \{Y_k\}_{k \geq 0}$ a set of formal sums such that $Y_k \, \mathcal{R}^{n}_{(M,N)} Z$ for all k and $Y_k \leq_{\texttt{apx}} Y_{k+1}$. As a consequence, there is a sequence Y'_k such that $Y_0 = Y'_0$ and $Y_{k+1} = Y_k + Y'_{k+1}$, i.e., $Y_k = \sum_{0 \leq h \leq k} Y'_h$. Hence, it follows from $Y = \sup S$ that $Y = \sum_{k \geq 0} Y'_k$.
The first item follows by the inductive hypothesis, since Y is the supremum of S and $Y_k \in S$ implies $\texttt{weight}(Y_k) \leq \texttt{weight}(Z)$.
As to the second item, we have that $Y \bullet C[M] = \sup\{Y_k \bullet C[M]\} = \sum_{k \geq 0} Y'_k \bullet C[M]$. We want to prove that if $\sum_{k \geq 0} Y'_k \bullet C[M] \Longmapsto X$ for some formal sum X then $X \, \mathcal{R}^{n+1}_{(M,N)} [\![Z \bullet C[N]]\!]$.
If $\sum_{k \geq 0} Y'_k \bullet C[M] \Longmapsto X$ then, by the definition of the multi-step reduction relation (which guarantees that only a finite number of terms are evaluated in the formal sum), there is an $m \geq 0$ such that $\sum_{0 \leq k \leq m} Y'_k \bullet C[M] \Longmapsto X'$ and $X = X' + \texttt{val}(\sum_{k > m} Y'_k \bullet C[M])$.
For any $m' \geq 0$ we have that

$$\textstyle\sum_{0 \leq k \leq m+m'} Y'_k \bullet C[M] \Longmapsto X' + \texttt{val}(\sum_{m \leq k \leq m+m'} Y'_k \bullet C[M])$$

Since $\sum_{0 \leq k \leq m+m'} Y'_k = Y_{m+m'}$ and $Y_{m+m'} \, \mathcal{R}^{n}_{(M,N)} Z$, by the inductive hypothesis we have $\sum_{0 \leq k \leq m+m'} Y'_k \bullet C[M] \Longmapsto X' + \texttt{val}(\sum_{m \leq k \leq m+m'} Y'_k \bullet C[M])$ implies

$$X' + \texttt{val}(\textstyle\sum_{m \leq k \leq m+m'} Y'_k \bullet C[M]) \; \mathcal{R}^{n}_{(M,N)} [\![Z \bullet C[N]]\!].$$

Hence, by the definition of $\mathcal{R}^{n+1}_{(M,N)}$ and by

$$X = \sup\{X' + \texttt{val}(\textstyle\sum_{m \leq k \leq m+m'} Y'_k \bullet C[M])\}_{m' \geq 0}$$

we derive that $X\,\mathcal{R}^{n+1}_{(M,N)}[\![Z\bullet C[N]]\!]$.

□

Then the result follows since $Y=\sum_i p_i;\lambda x.M_i\ \mathcal{S}_{\mathcal{E}}\ \sum_j q_j;\lambda x.N_j=Z$ implies $Y\ \mathcal{R}^n_{\mathcal{E}}\ Z$ for some n, and we have $\sum_i p_i\leq\sum_j q_j$ (by the first item of Lemma 8.56), and

$$\textstyle\sum_i p_i;[\![M_i\{P/x\}]\!]=\sup\{Y\mid\sum_i p_i;M_i\{P/x\}\Longmapsto Y\}\ \mathcal{R}^{n+1}_{\mathcal{E}}\ \sum_j q_j;[\![N_j\{Q/x\}]\!],$$

for all $P,Q\in\mathcal{E}^\star$ (by the second item of Lemma 8.56 and by the definition of relation $\mathcal{R}^{n+1}$), which implies $\mathcal{R}^{n+1}_{\mathcal{E}}\subseteq\mathcal{S}_{\mathcal{E}}$.

Proof of Lemma 8.18

Let $\mathcal{R}$ be a finite-step simulation saturated by approximants and let

$$\mathcal{S}=\{(C[M],C[N])\ \mid\ M\,\mathcal{R}\,N\}\cup\{((C[M],C[N]),Y,Z)\ \mid\ Y\ \mathcal{S}'_{(M,N)}\ Z\}$$

with

$$\begin{aligned}\mathcal{S}'=&\{((M,N),1;\lambda x.C'[M],1;\lambda x.C'[N])\mid M\ \mathcal{R}\ N\}\\ &\cup\{((M,N),Y,Z)\mid Y\ \mathcal{R}_{(M,N)}\ Z\}\\ &\cup\{((M,N),\emptyset,Z)\mid \text{ for some } M,N,Z\}\end{aligned}$$

We first prove the following result:

Lemma 8.57. *For any context C, if $M\ \mathcal{R}\ N$ and $C[M]\Longmapsto Y$ then $Y\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,[\![C[N]]\!]$.*

Proof. We prove by induction on the length n of the reduction that $M\,\mathcal{R}\,N$ and $C[M]\Longmapsto_n Y$ imply $Y\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,[\![C[N]]\!]$. If $Y=\emptyset$ then the result follows by the third set of $\mathcal{S}'$. Suppose that $Y\neq\emptyset$. If $n=0$ then we have two cases:

- $C=[\cdot]$ and M is a value. Then $M\Longmapsto Y$ and since $\mathcal{R}$ is a finite-step simulation we have that $Y\,\mathcal{R}_{(M,N)}[\![N]\!]=[\![C[N]]\!]$, which implies that they are in $\mathcal{S}'_{(M,N)}$ as well.

- $C=\lambda x.C'$. Then $Y=1;\lambda x.C'[M]\ \mathcal{S}'_{(M,N)}\ 1;\lambda x.C'[N]=[\![C[N]]\!]$, by the first set of $\mathcal{S}'$.

Suppose now that $C[M]\Longmapsto_{n+1}Y$.

- $C=[\cdot]$ and $M\Longmapsto_{n+1}Y$. The result follows from the fact that $\mathcal{R}$ is a finite-step simulation, as in the first case of $n=0$.

- $C=C_1\oplus C_2$ and $C[M]\longrightarrow\frac{1}{2};C_1[M]+\frac{1}{2};C_2[M]\Longmapsto_n Y$. Then $C_1[M]\Longmapsto_{n_1}Y_1$, $C_2[M]\Longmapsto_{n_2}Y_2$ and $Y=\frac{1}{2};Y_1+\frac{1}{2};Y_2$. We have $[\![C[N]]\!]=\frac{1}{2};[\![C_1[N]]\!]+\frac{1}{2};[\![C_2[N]]\!]$ and it follows from the inductive hypothesis on n_1 and n_2 that $Y_1\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,[\![C_1[N]]\!]$ and $Y_2\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,[\![C_2[N]]\!]$. Hence, $Y\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,[\![C[N]]\!]$.

- $C=C_1C_2$. Then $C[M]\Longmapsto_{n+1}Y$ implies that $C[M]\Longrightarrow_{n_1}Y_1C_2[M]\Longmapsto_{n_2}Y$, where $C_1[M]\Longmapsto_{n_1}Y_1=\sum_i p_i;\lambda x.P_i$. Since $n_1\leq n$ (by $Y\neq\emptyset$), we can apply the inductive hypothesis and derive that $Y_1\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,[\![C_1[N]]\!]$, i.e., $Y_1=\sum_j r_j{\cdot}Y'_j$ and $[\![C_1[N]]\!]=\sum_j r_j{\cdot}Z'_j$ for $Y'_j\ \mathcal{S}'_{(M,N)}\ Z'_j$. Hence, $Y_1C_2[M]\Longmapsto_{n_2}Y$

implies $Y'_jC_2[M] \Longmapsto_{n'_j} Y''_j$, with $\sum_j n'_j = n_2$ and $Y = \sum_j r_j{\cdot}Y''_j$, and $\llbracket C[N] \rrbracket = \llbracket \llbracket C_1[N] \rrbracket C_2[N] \rrbracket = \llbracket \sum_j r_j{\cdot}Z'_jC_2[N] \rrbracket = \sum_j r_j{\cdot}\llbracket Z'_jC_2[N] \rrbracket = \sum_j r_j; \llbracket Z'_j \bullet C_2[N] \rrbracket$.
Since $\mathtt{lift}(\,\mathtt{lift}(\,\mathcal{R}\,)\,) = \mathtt{lift}(\,\mathcal{R}\,)$ for any relation $\mathcal{R}$, the result follows if we can prove that for every j it holds that $Y'_jC_2[M] \Longmapsto_{n'_j} Y''_j$ implies $Y''_j\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,\llbracket Z'_j \bullet C_2[N] \rrbracket$. If $Y'_j\ \mathcal{S}'_{(M,N)}\ Z'_j$ then either $Y'_j = \emptyset$ and the result trivially follows or one of the following cases hold:

- $Y'_j = 1; \lambda x.C'[M]$ and $Z'_j = 1; \lambda x.C'[N]$. Hence, either $Y''_j = \emptyset$, in which case the result follows by the third set of $\mathcal{S}'$, or $Y'_jC_2[M] \longrightarrow 1; C'[M]\{{}^{C_2[M]}\!/\!_x\} \Longmapsto_{n''_j} Y''_j$, with $n''_j \leq n$. Terms $C'[M]\{{}^{C_2[M]}\!/\!_x\}$ and $C'[N]\{{}^{C_2[N]}\!/\!_x\}$ are respectively of the form $C''[M]$ and $C''[N]$, so we can apply the inductive hypothesis to derive that $Y''_j\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\, Z'_j \bullet C_2[N]$.
- $Y'_j\,\mathcal{R}_{\,(M,N)}Z'_j$. It is easy to check that $Y'_jC_2[M] \Longmapsto Y''_j$ iff $Y'''_j{\bullet}C_2[M] \Longmapsto Y''_j$ for some $Y'''_j \leq_{\mathtt{apx}} Y'_j$. Since $\mathcal{R}$ is finite-step simulation saturated by approximants we have that $Y'''_j{\bullet}C_2[M] \Longmapsto Y''_j$ implies $Y''_j\,\mathcal{R}_{\,(M,N)}\llbracket Z'_j \bullet C_2[N] \rrbracket$, and the result follows from $\mathcal{R}_{\,(M,N)} \subseteq \mathtt{lift}(\mathcal{S}'_{(M,N)})$.

□

We derive from Lemma 8.57 that $\mathcal{S}$ is a finite-step simulation up-to lifting as follows:

- Let $C[M]\ \mathcal{S}\ C[N]$ with $M\,\mathcal{R}\,N$. If $C[M] \Longmapsto Y$ then by Lemma 8.57 we have $Y\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,\llbracket C[N] \rrbracket$. Therefore, $Y\,\mathtt{lift}(\mathcal{S}_{(C[M],C[N])})\,\llbracket C[N] \rrbracket$.
- Let $1; \lambda x.C'[M]\ \mathcal{S}_{C[M],C[N]}\ 1; \lambda x.C'[N]$ with $M\,\mathcal{R}\,N$. Then for any C'' there is a context C''' such that $1; \lambda x.C'[M] \bullet C''[C[M]] = 1; C'''[M]$ and $1; \lambda x.C'[N] \bullet C''[C[N]] = 1; C'''[N]$. It is easy to check that $1; P \Longmapsto Y$ iff $P \Longmapsto Y$ and that $\llbracket 1; P \rrbracket = \llbracket P \rrbracket$ for any term P. Therefore, by Lemma 8.57 we derive that $1; C'''[M] \Longmapsto Y$ implies $Y\,\mathtt{lift}(\mathcal{S}'_{(M,N)})\,\llbracket 1; C'''[M] \rrbracket$, which implies that $Y\,\mathtt{lift}(\mathcal{S}_{C[M],C[N]})\,\llbracket 1; C'''[M] \rrbracket$.
- Let $Y\,\mathcal{R}_{\,C[M],C[N]}Z$ with $Y\,\mathcal{R}_{\,(M,N)}Z$. Then by the fact that $\mathcal{R}$ is a finite-step simulation it holds that for any C', $Y \bullet C'[C[M]] \Longmapsto Y'$ implies $Y'\,\mathcal{R}_{\,(M,N)}\llbracket Z \bullet C'[C[N]] \rrbracket$, which in turn implies $Y'\ \mathcal{S}_{C[M],C[N]}\ \llbracket Z \bullet C'[C[N]] \rrbracket$.
- Let $\emptyset\,\mathcal{R}_{\,C[M],C[N]}Z$. Then for any P we have that $\emptyset \bullet P \Longmapsto Y$ implies $Y = \emptyset$ and we stay in the third set.

Proof of Lemma 8.24

Let $\mathcal{R}$ be a finite-step simulation up-to lifting and context. We prove that

$$\begin{aligned}
\mathcal{R}' \overset{\mathsf{def}}{=}\ & \mathtt{Pairs}(\mathcal{R}) \\
& \cup \{((M,N), 1; \lambda x.C[M], 1; \lambda x.C[N]) \mid M\ \mathcal{R}\ N\} \\
& \cup \{((M,N), Y, Z) \mid Y'\,\mathcal{R}_{\,(M,N)}Z \text{ and } Y \leq_{\mathtt{apx}} Y', \text{ for some } Y'\} \\
& \cup \{((M,N), \emptyset, Z) \mid \text{ for some } M, N, Z\}
\end{aligned}$$

is a finite-step simulation up-to lifting, from which the result follows by $\mathcal{R} \subseteq \mathcal{R}'$.
Note that the relation $\mathtt{lift}(\mathcal{R}'_{(M,N)})$ is saturated by approximants, i.e., the following property holds: if $Y \leq_{\mathtt{apx}} Y'\,\mathtt{lift}(\mathcal{R}'_{(M,N)})\,Z$ then $Y\,\mathtt{lift}(\mathcal{R}'_{(M,N)})\,Z$. We first prove the following result:

Lemma 8.58. *For any context C, if $M \mathrel{\mathcal{R}} N$ and $C[M] \Longmapsto Y$ then $Y \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket C[N] \rrbracket$.*

Proof. We prove by induction on the number of small-step reductions n that $M \mathrel{\mathcal{R}} N$ and $C[M] \Longmapsto_n Y$ imply $Y \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket C[N] \rrbracket$. If $Y = \emptyset$ then the result holds by the last set of $\mathcal{R}'$. Suppose that $Y \neq \emptyset$. If $n = 0$ then we have two cases:

- $C = [\cdot]$ and M is a value. Then $M \Longmapsto Y$ and we have that $Y \mathrel{\texttt{lift}(\mathcal{R}_{(M,N)})} \llbracket N \rrbracket = \llbracket C[N] \rrbracket$.
- $C = \lambda x.C'$. Then $Y \leq_{\texttt{apx}} 1; \lambda x.C'[M] \texttt{dirac}((M,N)^\star) 1; \lambda x.C'[N] = \llbracket C[N] \rrbracket$.

Suppose now that $C[M] \Longmapsto_{n+1} Y$.

- $C = [\cdot]$ and $M \Longmapsto_{n+1} Y$. The result follows from the fact that $\mathcal{R}$ is a finite-step simulation up-to lifting and context, as in the first case of $n = 0$.
- $C = C_1 \oplus C_2$ and $C[M] \longrightarrow \frac{1}{2}; C_1[M] + \frac{1}{2}; C_2[N] \Longmapsto_n Y$. Then $C_1[M] \Longmapsto_{n_1} Y_1$, $C_2[N] \Longmapsto_{n_2} Y_2$ with $n_1 + n_2 \leq n$ and $Y = \frac{1}{2}; Y_1 + \frac{1}{2}; Y_2$. We have $\llbracket C[N] \rrbracket = \frac{1}{2}; \llbracket C_1[N] \rrbracket + \frac{1}{2}; \llbracket C_2[N] \rrbracket$ and it follows from the inductive hypothesis on n_1 and n_2 that $Y_1 \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket C_1[N] \rrbracket$ and $Y_2 \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket C_2[N] \rrbracket$. Hence, we derive $Y \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket C[N] \rrbracket$.
- $C = C_1 C_2$. Then $C[M] \Longmapsto_{n+1} Y$ implies that $C[M] \Longrightarrow_{n_1} Y_1 C_2[M] \Longmapsto_{n_2} Y$, where $C_1[M] \Longmapsto_{n_1} Y_1 = \sum_i p_i; \lambda x.P_i$. Since $n_1 \leq n$ (by $Y \neq \emptyset$), we can apply the inductive hypothesis and derive that $Y_1 \mathrel{\texttt{lift}(\mathcal{S}'_{(M,N)})} \llbracket C_1[N] \rrbracket$, i.e., $Y_1 = \sum_j r_j \cdot Y'_j$ and $\llbracket C_1[N] \rrbracket = \sum_j r_j \cdot Z'_j$ for $Y'_j \mathcal{R}'_{(M,N)} Z'_j$. Hence, $Y_1 C_2[M] \Longmapsto_{n_2} Y$ implies $Y'_j C_2[M] \Longmapsto_{n'_j} Y''_j$, with $\sum_j n'_j = n_2$ and $Y = \sum_j r_j \cdot Y''_j$, and $\llbracket C[N] \rrbracket = \llbracket \llbracket C_1[N] \rrbracket C_2[N] \rrbracket = \llbracket \sum_j r_j \cdot Z'_j C_2[N] \rrbracket = \sum_j r_j \cdot \llbracket Z'_j C_2[N] \rrbracket = \sum_j r_j; \llbracket Z'_j \bullet C_2[N] \rrbracket$. Since $\texttt{lift}(\texttt{lift}(\mathcal{R})) = \texttt{lift}(\mathcal{R})$ for any relation $\mathcal{R}$, the result follows if we can prove that for every j it holds that $Y'_j C_2[M] \Longmapsto_{n'_j} Y''_j$ implies $Y''_j \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket Z'_j \bullet C_2[N] \rrbracket$. If $Y'_j \mathrel{\mathcal{R}'_{(M,N)}} Z'_j$ then one of the following cases hold:
  - $Y'_j = 1; \lambda x.C'[M]$ and $Z'_j = 1; \lambda x.C'[N]$. Hence, either $Y''_j = \emptyset$, in which case the result follows by the last set of $\mathcal{R}'$, or $Y'_j C_2[M] \longrightarrow 1; C'[M]\{{}^{C_2[M]}/_{x}\} \Longmapsto_{n''_j} Y''_j$, with $n''_j \leq n$. Terms $C'[M]\{{}^{C_2[M]}/_{x}\}$ and $C'[N]\{{}^{C_2[N]}/_{x}\}$ are respectively of the form $C''[M]$ and $C''[N]$ and we can apply the inductive hypothesis to derive that $Y''_j \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket Z'_j \bullet C_2[N] \rrbracket$.
  - $Y'_j \leq_{\texttt{apx}} X \mathcal{R}_{(M,N)} Z'_j$. If $Y'_j C_2[M] \Longrightarrow Y''_j$ then there is a X' such that $X C_2[M] \Longmapsto_{n''_j} X'$ and $Y''_j \leq_{\texttt{apx}} X'$, for some $n''_j \leq n'_j$. It is easy to check that there is exists a X'' such that $X \bullet C_2[M] \Longmapsto_{n'''_j} X''$ and $X' \leq_{\texttt{apx}} X''$, for some $n'''_j < n''_j$. Since $X \mathcal{R}_{(M,N)} Z'_j$, there are two cases:
    * $X \bullet C_2[M] \Longrightarrow F$ and $Z'_j \bullet C_2[N] \Longrightarrow G$ with $F \mathrel{\texttt{lift}(\texttt{dirac}((M,N)^\star))} G$. Hence, $F \Longmapsto_{n''''_j} X'''$ with $X'' \leq_{\texttt{apx}} X'''$, for some $n''''_j \leq n'''_j < n$. We can apply the inductive hypothesis to the pairs in $(M,N)^\star$ whose projections respectively compose F and G through the lifting, and derive that $X''' \mathrel{\texttt{lift}(\texttt{lift}(\mathcal{R}'_{(M,N)}))} \llbracket G \rrbracket$. It follows from $Y''_j \leq_{\texttt{apx}} X'''$ that

$$Y''_j \mathrel{\texttt{lift}(\mathcal{R}'_{(M,N)})} \llbracket G \rrbracket = \llbracket Z'_j \bullet C_2[N] \rrbracket\,.$$

  - ∗ $X''\,\texttt{lift}(\texttt{dirac}((M,N)^{\star})\cup\mathcal{R}_{(M,N)})\,[\![Z'_j\bullet C_2[N]]\!]$.
    Since $Y''_j \leq_{\mathsf{apx}} X''$, this allows us to derive that $Y''_j\,\texttt{lift}(\mathcal{R}'_{M,N})\,[\![Z'_j\bullet C_2[N]]\!]$, since $\texttt{lift}(\mathcal{R}'_{(M,N)})$ is saturated by approximants.
- the result is immediate if $Y'_j=\emptyset$.

□

Then we derive from Lemma 8.58 that $\mathcal{R}'$ is a finite-step simulation up-to lifting:

1. if $M\;\mathcal{R}'\;N$ then $M\;\mathcal{R}\;N$, and $M\Longmapsto Y$ implies $Y\,\texttt{lift}(\mathcal{R}_{(M,N)})\,[\![N]\!]$, by the definition of $\mathcal{R}$. Then the result follows by $\mathcal{R}\subseteq\mathcal{R}'$.
2. if $Y\;\mathcal{R}'_{(M,N)}\;Z$ then:
   - if $Y\leq_{\mathsf{apx}} Y'\;\mathcal{R}_{(M,N)}\;Z$ then for all $P,Q\in(M,N)^{\star}$ we have two cases:
     - $Y'\bullet P\Longrightarrow F$ and $Z\bullet Q\Longrightarrow G$ with $F\,\texttt{lift}(\texttt{dirac}((M,N)^{\star}))\,G$.
       If $Y\bullet P\Longmapsto Y''$ then there is a Y''' such that $Y'\bullet P\Longrightarrow F\Longmapsto Y'''$ and $Y''\leq_{\mathsf{apx}} Y'''$. It follows from $F\,\texttt{lift}(\texttt{dirac}((M,N)^{\star}))\,G$ and from Lemma 8.58 that $F\Longmapsto Y'''$ implies $Y'''\,\texttt{lift}(\,\texttt{lift}(\mathcal{R}'_{(M,N)})\,)\,[\![G]\!]$, which is equivalent to saying that $Y'''\,\texttt{lift}(\mathcal{R}'_{(M,N)})\,[\![Z\bullet Q]\!]$. Since $F''\leq_{\mathsf{apx}} Y'''$ and $\texttt{lift}(\mathcal{R}'_{(M,N)})$ is saturated by approximants we derive $Y'''\,\texttt{lift}(\mathcal{R}'_{(M,N)})\,[\![Z\bullet Q]\!]$.
     - $Y'\bullet P\Longmapsto X$ and $X\,\texttt{lift}(\texttt{dirac}((M,N)^{\star})\cup\mathcal{R}_{(M,N)})\,[\![Z\bullet Q]\!]$.
       If $Y\bullet P\Longmapsto Y''$ then there is a Y''' such that $Y'\bullet P\Longmapsto Y''+Y'''=X$ and, by the definition of $\mathcal{R}$, $Y''+Y'''\,\texttt{lift}(\texttt{dirac}((M,N)^{\star})\cup\mathcal{R}_{(M,N)})\,[\![Z\bullet Q]\!]$, which implies $Y''+Y'''\,\texttt{lift}(\mathcal{R}'_{(M,N)})\,[\![Z\bullet Q]\!]$. Since $Y''\leq_{\mathsf{apx}} Y''+Y'''$, we have $Y''\,\texttt{lift}(\mathcal{R}'_{(M,N)})\,[\![Z\bullet Q]\!]$.
   - if $Y=1;\lambda x.C[M]$ and $Z=1;\lambda x.C[N]$ with $M\;\mathcal{R}\;N$ then for all $P,Q\in(M,N)^{\star}$ we have $Y\bullet P=1;C'[M]$ and $Z\bullet Q=1;C'[N]$ for some C'. Then by Lemma 8.58 we have that $Y\bullet P\Longmapsto Y'$ implies $Y'\,\texttt{lift}(\mathcal{R}'_{(M,N)})\,[\![Z\bullet Q]\!]$.
   - if $Y=\emptyset$ then the simulation clause holds and we stay in the last set of $\mathcal{R}'$.

Proof of Lemma 8.28

Let $\rhd^{\star t}$ denote the transitive closure of $\rhd^{\star}$, i.e., $P\rhd^{\star t}P'$ if there are $P=P_1,P_2,...,P_k=P'$ such that for every $1\leq i<k$ there are a context C_i and tuples $\widetilde{P_i},\widetilde{P'_i}$ with $P_i=C_i[\widetilde{P_i}]$ and $P_{i+1}=C_i[\widetilde{P'_i}]=C_{i+1}[\widetilde{P_{i+1}}]$ and $\widetilde{P_i}\rhd\widetilde{P'_i}$.

Define the following term relation

$$\mathcal{R}_{(M,N)}\overset{\mathsf{def}}{=}(\rhd^{\widehat{\star}^{t}}\;(M,N)^{\widehat{\star}})\cup(\rhd^{\widehat{\star}^{t}}\mathcal{T}_{(M,N)})$$

Let $\texttt{liftd}()$ denote $\texttt{lift}(\texttt{dirac}())$.

Lemma 8.59. *We prove that*

(*) $P\rhd^{\star t}P'$ *and* $P\Longmapsto_n Y$ *imply* $P'\Longmapsto_{n'}Y'$ *with* $n\geq n'$ *and* $Y\,\texttt{liftd}(\rhd^{\widehat{\star}^{t}})\leq_{\mathsf{apx}} Y'$

(**) $P(\rhd^{\star t}\;(M,N)^{\star})Q$ *and* $P\Longmapsto Y$ *imply* $Y\,\texttt{disliftd}(\mathcal{R}_{(M,N)})\leq_{\mathsf{apx}}[\![Q]\!]$

Proof. We first prove (*). Let $P = P_1, P_2, ..., P_k = P'$ be such that for every i, for $1 \leq i < k$, there are a context C_i and tuples $\widetilde{P_i}, \widetilde{P'_i}$ with $P_i = C_i[\widetilde{P_i}]$ and $P_{i+1} = C_i[\widetilde{P'_i}] = C_{i+1}[\widetilde{P}_{i+1}]$ and $\widetilde{P_i} \rhd \widetilde{P'_i}$. Suppose that $P \Longmapsto_n Y$. We prove by induction on k that for every i such that $1 \leq i < k$ we have $P_i \Longmapsto_{m_i} Y_i$ and $m_i \geq m_{i+1}$ and $Y_i\,\texttt{liftd}(\rhd^{\widehat{\star}^t}) \leq_{\texttt{apx}} Y_{i+1}$.
The result is trivial for $k = 1$. Suppose that $k = k' + 1$. The result follows from the inductive hypothesis and from

$$Y\,\texttt{liftd}(\rhd^{\widehat{\star}^t}) \leq_{\texttt{apx}} Y'\,\texttt{liftd}(\rhd^{\widehat{\star}^t}) \leq_{\texttt{apx}} Y'' \text{ implies } Y\,\texttt{liftd}(\rhd^{\widehat{\star}^t}) \leq_{\texttt{apx}} Y'',$$

if we can prove that for any C and $\widetilde{P}, \widetilde{P}'$ such that $\widetilde{P} \rhd \widetilde{P}'$ we have $C[\widetilde{P}] \Longmapsto_m Y$ implies $C[\widetilde{P}'] \Longmapsto_{m'} Y'$ with $m \geq m'$ and $Y\,\texttt{liftd}(\rhd^{\widehat{\star}^t}) \leq_{\texttt{apx}} Y'$. This is proved by induction on m. If $m = 0$ then $Y = Y'$ and the result follows. Suppose $C[\widetilde{P}] \Longmapsto_{m+1} Y$. We have three cases:

- if $C = [\cdot]$ then the result immediately follows by $\widetilde{P} = P \rhd P' = \widetilde{P}'$.

- $C = C_1 \oplus C_2$.
  Then there are Y_1, Y_2 such that $Y = \frac{1}{2}; Y_1 + \frac{1}{2}; Y_2$ and $C_i[\widetilde{P}] \Longrightarrow_{m_i} Y_i$ for $i = 1, 2$ and $m_i < m + 1$. The result follows from the inductive hypothesis

- $C = C_1C_2$.
  If $C_1[\widetilde{P}]$ is a value then, by the definition of $\rhd$, $C_1[\widetilde{P}']$ is exactly the same value. Then the terms resulting after the β-reduction are in $\rhd^{\widehat{\star}^t}$, and we conclude by the inductive hypothesis.
  If $C_1[\widetilde{P}] \Longmapsto_{m_1} Y_1$ then we can apply the inductive hypothesis to m_i and the result follows.

We can now prove (**) by showing by induction on n that $P(\rhd^{\star t}\ (M, N)^{\star})Q$ and $P \Longmapsto_n Y$ imply $Y\texttt{disliftd}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} \llbracket Q \rrbracket$.
Let $P \rhd^{\star t} P'$, $P' = C[M]$ and $Q = C[N]$. If $n = 0$ and $Y \neq \emptyset$ then P is a value, $Y = 1; P$ and by (*) P' is a value too, with $P \rhd^{\widehat{\star}^t} P'$. We have two cases:

- $C = [\cdot]$, with $P' = M$ a value, and $Q = N$.
  By the definition of $\mathcal{T}_{(M,N)}$ we have $1; P' = \llbracket M \rrbracket\texttt{disliftd}(\mathcal{T}_{(M,N)})\llbracket N \rrbracket$ and it follows from $P \rhd^{\widehat{\star}^t} P'$ that $Y\texttt{disliftd}(\rhd^{\widehat{\star}^t}\mathcal{T}_{(M,N)})\llbracket Q \rrbracket$.

- $P' = \lambda x.C'[M]$ and $Q = \lambda x.C'[N]$.
  Then $Y\texttt{disliftd}(\rhd^{\widehat{\star}^t}\ (M, N)^{\widehat{\star}})\llbracket Q \rrbracket$.

If $P \Longmapsto_{n+1} Y$ then by (*) we have that $P' \Longmapsto_m Y'$ with $m \leq n{+}1$ and $Y\,\texttt{liftd}(\rhd^{\widehat{\star}^t}) \leq_{\texttt{apx}} Y'$. Suppose that $Y' \neq \emptyset$. We have three cases:

- $C = [\cdot]$. If $M \Longmapsto_m Y'$ then $Y' \leq_{\texttt{apx}} \llbracket M \rrbracket\texttt{disliftd}(\mathcal{T}_{(M,N)})\llbracket N \rrbracket$, which implies

  $$Y\texttt{disliftd}(\mathcal{T}_{(M,N)}) \leq_{\texttt{apx}} \llbracket N \rrbracket$$

  by $\leq_{\texttt{apx}}\ \texttt{disliftd}(\mathcal{T}_{(M,N)}) \leq_{\texttt{apx}} \subseteq\ \texttt{disliftd}(\mathcal{T}_{(M,N)}) \leq_{\texttt{apx}}$. Since for any $\mathcal{S}, \mathcal{S}'$ we have $\texttt{liftd}(\mathcal{S})\,\texttt{disliftd}(\mathcal{S}') \subseteq \texttt{disliftd}(\mathcal{S}\mathcal{S}')$, we derive

  $$Y\texttt{disliftd}(\rhd^{\widehat{\star}^t}\mathcal{T}_{(M,N)}) \leq_{\texttt{apx}} \llbracket N \rrbracket\,.$$

- $C = C_1 + C_2$. Then there are Y_1, Y_2 such that $Y' = \frac{1}{2}; Y_1 + \frac{1}{2}; Y_2$ and $C_i[M] \Longrightarrow_{m_i} Y_i$ for $i = 1, 2$ and $m_i \leq n$. By the inductive hypothesis, we have that

  $$Y' \, \texttt{liftd}(\texttt{dislift}(\mathcal{R}_{(M,N)})) \leq_{\texttt{apx}} [\![Q]\!] \, .$$

  Hence, since for any pari of relations $\mathcal{S}, \mathcal{S}'$ we have $\texttt{liftd}(\texttt{disliftd}(\cdot)) = \texttt{disliftd}(\cdot)$, and $\texttt{liftd}(\mathcal{S}) \, \texttt{disliftd}(\mathcal{S}') \subseteq \texttt{disliftd}(\mathcal{S}\mathcal{S}')$, and by $\rhd^{\star t} \rhd^{\star t} = \rhd^{\star t}$, we derive

  $$Y \texttt{dislift}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} [\![Q]\!] \, .$$

- $C = C_1 C_2$. We consider two cases:
  - $C_1[M] = M$ and $M = \lambda x.M'$ and $N = \lambda x.N'$ are values.
    Then $M \ \mathcal{T}_{(M,N)} \ N$ and we have $M C_2[M] \Longmapsto_m Y'$ iff $M'\{C_2[M]/x\} \overset{\text{d}}{\Longrightarrow} F \Longmapsto_{m'} Y'$, with $m' < m \leq n + 1$ and $[\![N C_2[N]]\!] = [\![N'\{C_2[N]/x\}]\!] = [\![G]\!]$, where $F \texttt{disliftd}(\rhd^{\star} \mathcal{T}^{\star -}_{(M,N)}) G$ (note that here the determinism of the reduction to F is used in order to guarantee that $F \Longmapsto_{m'} Y'$ and that $m' < m \leq n + 1$). Hence, we have $Y' \texttt{dislift}(\{(Y_i, Z_i)\}_i) [\![G]\!]$, with either $Y_i \texttt{dirac}(\rhd^{\widehat{\star}} \mathcal{T}^{\star -}_{(M,N)}) Z_i$ (which implies $Y_i \texttt{dirac}(\mathcal{R}_{(M,N)}) Z_i$) or $M_i \Longmapsto_{m_i} Y_i$ and $[\![N_i]\!] = Z_i$ for $M_i \rhd^{\star} (M,N)^{\star} N_i$ and $m_i \leq m' < n + 1$, and by applying the inductive hypothesis we derive $Y_i \texttt{disliftd}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} Z_i$. Therefore, $Y \, \texttt{liftd}(\rhd^{\star t}) \leq_{\texttt{apx}} Y' \texttt{disliftd}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} [\![Q]\!]$, and the result follows.
  - $C_1[M] = \lambda x.C_1'[M]$ and $C_1[N] = \lambda x.C_1'[N]$.
    We can apply the inductive hypothesis to $C_1'[M]\{C_2[M]/x\} \Longmapsto_{m'} Y'$ to derive

    $$Y' \texttt{disliftd}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} [\![C'[N]\{C_2[N]/x\}]\!],$$

    then we have $Y \, \texttt{liftd}(\rhd^{\star t}) \leq_{\texttt{apx}} Y' \texttt{disliftd}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} [\![C'[N]\{C_2[N]/x\}]\!]$, which implies the result.

□

We can now derive that the relation

$$\mathcal{S} = \{(M,N)\} \cup \{((M,N), 1; V, 1; W) \mid V \ \mathcal{R}_{(M,N)} \ W\} \cup \{((M,N), \emptyset, Z) \mid \text{ for any } Z\}$$

is a finite-step simulation up-to distribution and lifting:

- since $M (\rhd^{\star t} \ (M,N)^{\star}) N$, by Lemma 8.59(**) we have that if $M \Longmapsto Y$ then

  $$Y \texttt{disliftd}(\mathcal{R}_{(M,N)}) Y' \leq_{\texttt{apx}} [\![N]\!],$$

  which implies $Y \texttt{dislift}(\mathcal{S}_{(M,N)}) [\![N]\!]$ (using the last set of relation $\mathcal{S}_{(M,N)}$ to eliminate the approximation preorder $\leq_{\texttt{apx}}$).

- the weight of the formal sums is the same.

- if $1; V \ \mathcal{S}_{(M,N)} \ 1; W$ and $V \rhd^{\widehat{\star} t} \ V'(M,N)^{\widehat{\star}} W$ then for any $(P,Q) \in (M,N)^{\star}$ we have that $VP \rhd^{\star t} \ V'P(M,N)^{\star} WQ$. Then clause (2b) of finite-step simulation holds by Lemma 8.59(**).

If $V \rhd^{\widehat{\star}^t} V' \mathcal{T}_{(M,N)} W$ then for any $(P,Q) \in (M,N)^\star$ we have that $VP \rhd^{\star t} V'P$. Hence, $VP \Longmapsto Y$ implies by Lemma 8.59(*) that $V'P \Longmapsto Y'$ with $Y\, \texttt{liftd}(\rhd^{\widehat{\star}^t}) \leq_{\texttt{apx}} Y'$. By the definition of $\mathcal{T}_{(M,N)}$ and by the fact that $\overset{\mathrm{d}}{\Longrightarrow}$ is deterministic, we derive from $V'P \Longmapsto Y'$ and from Lemma 8.59(**) that $Y'\texttt{disliftd}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} [\![WQ]\!]$ (see the proof of Lemma 8.59(**), application case, for more details). Then $Y\texttt{disliftd}(\mathcal{R}_{(M,N)}) \leq_{\texttt{apx}} [\![WQ]\!]$, which implies $Y\texttt{disliftd}(\mathcal{S}_{(M,N)})[\![WQ]\!]$.

Proof of Theorem 8.51

Let $\mathcal{R}$ be a finite-step $\{\widetilde{l}\}$-simulation saturated by approximants. We first define, for any pair of terms (M,N), the preorder $\leq_{\texttt{cce}(M,N)}$ (context closure of environments) on pairs of environment formal sums with store: $(\mathbf{Y},\mathbf{Z}) \leq_{\texttt{cce}(M,N)} (\mathbf{Y}',\mathbf{Z}')$ if $\mathbf{Y} = \sum_i p_i; s_i; \widetilde{V}_i, \mathbf{Z} = \sum_j q_j; t_j; \widetilde{W}_j$ with $\mid \mathbf{Y} \mid = \mid \mathbf{Z} \mid$ and $\mathbf{Y}' = \sum_i p_i; s_i; \widetilde{V}'_i, \mathbf{Z}' = \sum_j q_j; t_j; \widetilde{W}'_j$ with $\mid \mathbf{Y}' \mid = \mid \mathbf{Z}' \mid$ and

- for every index r in $\mid \mathbf{Y} \mid$ there is an index r' in $\mid \mathbf{Y}' \mid$ such that $\mathbf{Y}\lfloor_r = \mathbf{Y}'\lfloor_{r'}$ and $\mathbf{Z}\lfloor_r = \mathbf{Z}'\lfloor_{r'}$;
- for every index $r' \leq\mid \mathbf{Y}' \mid$ there is a location-free value-context C such that for every i,j it holds that $r'(\widetilde{V}'_i) = C[M,\widetilde{V}_i]$ and $r'(\widetilde{W}'_j) = C[N,\widetilde{W}_j]$ (i.e., $(\mathbf{Y}'\lfloor_{r'}, \mathbf{Z}'\lfloor_{r'}) \in (\{(M,\widetilde{V}_i\}_i, \{N,\widetilde{W}_j\}_j)^{\widehat{\star}})$.

For any indexed relation $\mathcal{R}_{(M,N)}$ on environment formal sums, we write $\mathcal{R}^{\texttt{cce}}_{(M,N)}$ for relation $\leq_{\texttt{cce}(M,N)} (\mathcal{R}_{M,N})$, i.e.,

$$\mathcal{R}^{\texttt{cce}}_{(M,N)} = \{(\mathbf{Y},\mathbf{Z}) \mid \exists \mathbf{Y}',\mathbf{Z}' \text{ such that } (\mathbf{Y}',\mathbf{Z}') \leq_{\texttt{cce}(M,N)} (\mathbf{Y},\mathbf{Z}) \ \wedge\ \mathbf{Y}' \mathcal{R}_{(M,N)} \mathbf{Z}'\}$$

Note that we can add to a finite-step $\{\widetilde{l}\}$-simulation $\mathcal{R}$ saturated by approximants the set

$$\{((M,N),\emptyset,\mathbf{Y}) \mid \mathbf{Y} \text{ is an environment formal sum}\}$$

for any (M,N), and we still have that $\mathcal{R}$ is a finite-step $\{\widetilde{l}\}$-simulation saturated by approximants, since the presence of the empty formal sum on the left is irrelevant and trivially satisfies the simulation clauses. Hence, we assume that finite-step simulations have this property, that we refer to as *$\mathcal{R}$ is saturated by $\emptyset$*.

The following result is used in the proof of Lemma 8.61 (in Lemma 8.60 and 8.61, the set of locations parametrizing the relations is not relevant, hence we omit it).

Lemma 8.60. *Let $\mathcal{R}$ be a finite-step simulation and let*

$$\mathbf{Y} = \textstyle\sum_i p_i; s_i; \widetilde{V}_i\ \mathcal{R}^{\texttt{cce}}_{(M,N)} \sum_j q_j; t_j; \widetilde{W}_j = \mathbf{Z}$$

Let $(\{\lambda x.P_i\}_i, \{\lambda x.Q_j\}_j) \in (\{M,\widetilde{V}_i\}_i, \{N,\widetilde{W}_j\}_j)^{\widehat{\star}}$ and $(\{T_i\}_i, \{U_j\}_j) \in (\{M,\widetilde{V}_i\}_i, \{N,\widetilde{W}_j\}_j)^{\widehat{\star}}$. Then one of the following holds:

- $(\{P_i\{{}^{T_i}\!/\!_i\}\}_i, \{Q_j\{{}^{U_j}\!/\!_x\}\}_j) \in (\{M,\widetilde{V}_i\}_i, \{N,\widetilde{W}_j\}_j)^\star$
- *for every* $\mathbf{W}$, $\sum_i p_i; s_i; \widetilde{V}_i, \lambda x.P_i; P_i\{{}^{T_i}\!/\!_i\} \Longmapsto \mathbf{W}$ *implies*

$$\mathbf{W}\,\texttt{lift}(\mathcal{R}^{\texttt{cce}}_{(M,N)})\, [\![\textstyle\sum_j q_j; t_j; \widetilde{W}_j, \lambda x.Q_j; Q_j\{{}^{U_j}\!/\!_x\}]\!]$$

Proof. We have three cases:

- if $\lambda x.P_i = \lambda x.C'[M, \widetilde{V_i}]$ and $\lambda x.Q_j = \lambda x.C'[N, \widetilde{W_j}]$ for some location-free context C', then there is a location-free context C'' such that

$$\begin{array}{c}\sum_i p_i; s_i; \widetilde{V_i}; P_i\{T_i/x\} = \sum_i p_i; s_i; \widetilde{V_i}; C''[M, \widetilde{V_i}] \\ \sum_j q_j; t_j; \widetilde{W_j}; Q_j\{U_j/x\} = \sum_j q_j; t_j; \widetilde{W_j}; C''[N, \widetilde{W_j}]\end{array}$$

  and the first item holds.

- if there is an r such that $\lambda x.P_i = (\widetilde{V'_i})_r$ and $\lambda x.Q_j = (\widetilde{W'_j})_r$ for some

$$\mathbf{Y}' = \sum_i p_i; s_i; \widetilde{V'_i} \; \mathcal{R}_{(M,N)} \sum_j q_j; t_j; \widetilde{W'_j} = \mathbf{Z}'$$

  such that $(\mathbf{Y}', \mathbf{Z}') \leq_{\mathtt{cce}(M,N)} (\mathbf{Y}, \mathbf{Z})$, then it follows from the fact that $\mathcal{R}$ is a finite-step simulation that $\sum_i p_i; s_i; \widetilde{V'_i}; P_i\{T_i/x\} \Longmapsto \mathbf{W}'$ implies

$$\mathbf{W}' \, \mathtt{lift}(\mathcal{R}_{(M,N)}) \, \llbracket \textstyle\sum_j q_j; t_j; \widetilde{W'_j}; Q_j\{U_j/x\} \rrbracket.$$

  Since $\sum_i p_i; s_i; \widetilde{V_i}, \lambda x.P_i; P_i\{T_i/i\} \Longmapsto \mathbf{W}$ implies

$$(\mathbf{W}', \llbracket \textstyle\sum_j q_j; t_j; \widetilde{W'_j}; Q_j\{U_j/x\} \rrbracket) \leq_{\mathtt{cce}(M,N)} (\mathbf{W}, \llbracket \textstyle\sum_j q_j; t_j; \widetilde{W_j}, \lambda x.Q_j; Q_j\{U_j/x\} \rrbracket),$$

  we derive

$$\mathbf{W} \, \mathtt{lift}(\mathcal{R}^{\mathtt{cce}}_{(M,N)}) \, \llbracket \textstyle\sum_j q_j; t_j; \widetilde{W_j}, \lambda x.Q_j; Q_j\{U_j/x\} \rrbracket$$

- if $\lambda x.P_i = M$ and $\lambda x.Q_j = N$ then M and N are values and by clause (2g) applied to $\mathcal{R}_{(M,N)}$ we have

$$\textstyle\sum_i p_i; s_i; \widetilde{V'_i}, M \; \mathcal{R}_{(M,N)} \sum_j q_j; t_j; \widetilde{W'_j}, N$$

  for some $(\sum_i p_i; s_i; \widetilde{V'_i}, \sum_j q_j; t_j; \widetilde{W'_j}) \leq_{\mathtt{cce}(M,N)} (\mathbf{Y}, \mathbf{Z})$. By clause (2b) applied to $\mathcal{R}_{(M,N)}$ we have $\sum_i p_i; s_i; \widetilde{V'_i}, M; P_i\{T_i/x\} \Longmapsto \mathbf{W}'$ implies

$$\mathbf{W}' \, \mathtt{lift}(\mathcal{R}_{(M,N)}) \, \llbracket \textstyle\sum_j q_j; t_j; \widetilde{W'_j}, N; Q_j\{U_j/x\} \rrbracket \, .$$

  Since by assumption $\lambda x.P_i = M$ and $\lambda x.Q_j = N$, there are already columns in the dynamic environments of $\mathbf{Y}$ and $\mathbf{Z}$ composed by M and N respectively, and thus

$$(\mathbf{W}', \llbracket \textstyle\sum_j q_j; t_j; \widetilde{W'_j}, N; Q_j\{U_j/x\} \rrbracket) \leq_{\mathtt{cce}(M,N)} \geq_{\mathtt{env}} (\mathbf{W}, \llbracket \textstyle\sum_j q_j; t_j; \widetilde{W_j}; Q_j\{U_j/x\} \rrbracket),$$

  from which the result follows.

□

In what follows, we sometimes use relations $\leq_{\mathtt{lift}}$ and $\leq_{\mathtt{lift}} \leq_{\mathtt{env}}$, defined as follows:

- $(\mathbf{Y}, \mathbf{Z}) \leq_{\mathtt{lift}} \{(\mathbf{Y}_g, \mathbf{Z}_g)\}_g$ if there are probability values p_g such that $\mathbf{Y} = \sum_g p_g \cdot \mathbf{Y}_g$ and $\mathbf{Z} = \sum_g p_g \cdot \mathbf{Z}_g$;

- $(\mathbf{Y},\mathbf{Z}) \leq_{\texttt{lift}}\leq_{\texttt{env}} \{(\mathbf{Y}_g,\mathbf{Z}_g)\}_g$ if there are probability values p_g such that $\mathbf{Y} = \sum_g p_g \cdot \mathbf{Y}'_g$ and $\mathbf{Z} = \sum_g p_g \cdot \mathbf{Z}'_g$ with $(\mathbf{Y}'_g,\mathbf{Z}'_g) \leq_{\texttt{env}} (\mathbf{Y}_g,\mathbf{Z}_g)$ for every g.

The notation, as for relations $\leq_{\texttt{env}}$ and $\leq_{\texttt{cce}(M,N)}$, is extended to environment formal sums with running terms by requiring the running term to be the same everywhere.

Lemma 8.61. *Suppose that $\mathcal{R}_{(M,N)}$ is a finite-step simulation saturated by approximants (only defined on formal sums). For any location-free context C if $\mathbf{Y}\ \mathcal{R}^{\texttt{cce}}_{(M,N)}\ \mathbf{Z}$ and $\mathbf{Y}; C[M,\mathbf{Y}] \mathrel{|}\!\!\Longrightarrow \mathbf{W}$ then $\mathbf{W}\,\texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}^{\texttt{cce}}_{(M,N)}))\,\llbracket \mathbf{Z}; C[N,\mathbf{Z}]\rrbracket$.*

Proof. Suppose that $\mathbf{Y} = \sum_i p_i; s_i; \widetilde{V_i}\ \mathcal{R}^{\texttt{cce}}_{(M,N)}\ \sum_j q_j; t_j; \widetilde{W_j} = \mathbf{Z}$, with $\mathbf{Y}' = \sum_i p_i; s_i; \widetilde{V'_i}$ and $\mathbf{Z}' = \sum_j q_j; t_j; \widetilde{W'_j}$ related by $\mathcal{R}_{(M,N)}$ and such that $(\mathbf{Y}',\mathbf{Z}') \leq_{\texttt{cce}(M,N)} (\mathbf{Y},\mathbf{Z})$.

We prove by induction on n and then by induction on the structure of C that

$\sum_i p_i; s_i; \widetilde{V_i}; C[M,\widetilde{V_i}] \mathrel{|}\!\!\Longrightarrow_n \mathbf{W}$ implies $\mathbf{W}\,\texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}^{\texttt{cce}}_{(M,N)}))\,\llbracket \sum_j q_j; t_j; \widetilde{W_j}; C[N,\widetilde{W_j}]\rrbracket$.

(In the proof we sometimes assume that $\widetilde{V_i} = \widetilde{V'_i}, \widetilde{V''_i}$ and $\widetilde{W_j} = \widetilde{W'_j}, \widetilde{W''_j}$. This does not affect the results since the context closure of environments allows to permute the columns in the formal sums.)

If $\mathbf{W} = \emptyset$ then the result follows by the fact that $\mathcal{R}$ is saturated by $\emptyset$. Suppose that $\mathbf{W} \neq \emptyset$ and $n = 0$. We have one of the following cases:

- $C = [\cdot]_1$ and M is a value. Then

  $$\textstyle\sum_i p_i; \widetilde{V'_i} \cdot (1; \langle s_i\, ;\, ; \rangle M)\,\texttt{lift}(\mathcal{R}_{(M,N)})\, \sum_j q_j; \widetilde{W'_j} \cdot \llbracket \langle t_j\, ;\, N\rangle \rrbracket$$

  and we derive

  $$\textstyle\sum_i p_i; \widetilde{V_i} \cdot (1; \langle s_i\, ;\, ; \rangle M)(\,\texttt{lift}(\mathcal{R}^{\texttt{cce}}_{(M,N)})\,) \sum_j q_j; \widetilde{W_j} \cdot \llbracket \langle t_j\, ;\, N\rangle \rrbracket\,.$$

- $C \neq [\cdot]_1$ and for every i, j, $C[M,\widetilde{V_i}]$ and $C[N,\widetilde{W_j}]$ are values. Then it follows from the definition of $\leq_{\texttt{cce}(M,N)}$ that

  $$\textstyle\mathbf{W} = \sum_i p_i; s_i; \widetilde{V_i}, C[M,\widetilde{V_i}]\ \mathcal{R}^{\texttt{cce}}_{(M,N)}\ \sum_j q_j; t_j; \widetilde{W_j}, C[N,\widetilde{W_j}] = \llbracket \sum_j q_j; t_j; \widetilde{W_j}, C[N,\widetilde{W_j}]\rrbracket\,.$$

Suppose now that $\sum_i p_i; s_i; \widetilde{V_i}; C[M,\widetilde{V_i}] \mathrel{|}\!\!\Longrightarrow_{n+1} \mathbf{W}$, with $\mathbf{W} \neq \emptyset$. We have the following cases:

- $C = [\cdot]_1$ and M is not a value. Then the result follows analogously to the $n = 0$ case.

- $C = C_1 \oplus C_2$. We have that $\sum_i p_i; s_i; \widetilde{V_i}; (C_1 \oplus C_2)[M,\widetilde{V_i}] \mathrel{|}\!\!\Longrightarrow \mathbf{W}$ implies

  $$\textstyle\sum_i p_i; s_i; \widetilde{V_i}; (C_1 \oplus C_2)[M,\widetilde{V_i}] \longrightarrow \frac{1}{2}\cdot \sum_i p_i; s_i; \widetilde{V_i}; C_1[M,\widetilde{V_i}] + \frac{1}{2}\sum_i p_i; s_i; \widetilde{V_i}; C_2[M,\widetilde{V_i}]$$

  and $\mathbf{W} = \frac{1}{2}\cdot \mathbf{W}_1 + \frac{1}{2}\cdot \mathbf{W}_2$, with $\sum_i p_i; s_i; \widetilde{V_i}; C_h[M,\widetilde{V_i}] \mathrel{|}\!\!\Longrightarrow_{n_h} \mathbf{W}_h$, for $h = 1, 2$ and $n_h \leq n$. Since

  $$\textstyle\llbracket \sum_j q_j; t_j; \widetilde{W_j}; (C_1 \oplus C_2)[N,\widetilde{W_j}]\rrbracket =$$

$$\tfrac{1}{2} \cdot [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; C_1[N, \widetilde{W_j}]]\!] + \tfrac{1}{2} \cdot [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; C_1[N, \widetilde{W_j}]]\!]$$

we can apply the inductive hypothesis to n_1 and n_2 and derive

$$\mathbf{W}\, \mathtt{lift}(\,\mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}^{\mathtt{cce}}_{(M,N)}))\,)\, [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; (C_1 \oplus C_2)[N, \widetilde{W_j}]]\!]$$

The result follows from $\mathtt{lift}(\,\mathtt{lift}(\mathcal{R})\,) = \mathtt{lift}(\mathcal{R})$.

- $C = C_1C_2$.
  Suppose that there exists some i such that $C_1[M, \widetilde{V_i}]$ are not values and they perform some small steps in the reduction leading to $\mathbf{W}$. Hence, there is a set $I' \subseteq I$ such that

$$\textstyle\sum_{i \in I} p_i; s_i; \widetilde{V_i}; C_1[M, \widetilde{V_i}]C_2[M, \widetilde{V_i}] \Longrightarrow_{n_1} \sum_{i \in I' \subseteq I, k} p_{i,k}; s_{i,k}; \widetilde{V_i}; V_{i,k}C_2[M, \widetilde{V_i}] \mathrel{|}\!\!\Longrightarrow_{n_2} \mathbf{W}$$

with $n_1 > 1$ and

$$\textstyle\sum_{i \in I} p_i; s_i; \widetilde{V_i}; C_1[M, \widetilde{V_i}] \mathrel{|}\!\!\Longrightarrow_{n_1} \sum_{i \in I', k} p_{i,k}; s_{i,k}; \widetilde{V_i}, V_{i,k}\,.$$

We have that

$$[\![\textstyle\sum_{j \in J} q_j; t_j; \widetilde{W_j}; C_1[N, \widetilde{W_j}]C_2[N, \widetilde{W_j}]]\!] = [\![\sum_{j \in J' \subseteq J, h} q_{j,h}; t_{j,h}; \widetilde{W_j}; W_{j,h}C_2[N, \widetilde{W_j}]]\!]$$

for $[\![\sum_{j \in J} q_j; t_j; \widetilde{W_j}; C_1[N, \widetilde{W_j}]]\!] = \sum_{j \in J', h} q_{j,h}; t_{j,h}; \widetilde{W_j}, W_{j,h}$. Then we can apply the inductive hypothesis to n_1 and derive that

$$\textstyle\sum_{i \in I', k} p_{i,k}; s_{i,k}; \widetilde{V_i}, V_{i,k}\, \mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}^{\mathtt{cce}}_{(M,N)}))\, \sum_{j \in J', h} q_{j,h}; t_{j,h}; \widetilde{W_j}, W_{j,h}$$

which is equivalent to saying that

$$(\textstyle\sum_{i \in I', k} p_{i,k}; s_{i,k}; \widetilde{V_i}, V_{i,k}, \sum_{j \in J', h} q_{j,h}; t_{j,h}; \widetilde{W_j}, W_{j,h}) \leq_{\mathtt{lift}} \leq_{\mathtt{env}} \{(\mathbf{Y}_g, \mathbf{Z}_g)\}_g \subseteq \mathcal{R}^{\mathtt{cce}}_{(M,N)}$$

with $\mathbf{Y}_g = \sum_{i,k \in I_g} p'_{i,k}; s_{i,k}; \widetilde{V}_{i,k}$, and $\mathbf{Z}_g = \sum_{j,h \in J_g} q'_{j,h}; t_{j,h}; \widetilde{W}_{j,h}$ such that for every g there is a context C_g such that for all $i, k \in I_g$ and for all $j, h \in J_g$ we have $V_{i,k}C_2[M, \widetilde{V_i}] = C_g[M, \widetilde{V}_{i,k}]$ and $W_{j,h}C_2[N, \widetilde{W_j}] = C_g[N, \widetilde{W}_{j,h}]$.
If $\sum_{i \in I' \subseteq I, k} p_{i,k}; s_{i,k}; \widetilde{V_i}; V_{i,k}C_2[M, \widetilde{V_i}] \mathrel{|}\!\!\Longrightarrow_{n_2} \mathbf{W}$ then for every g there is a $\mathbf{W}_g$ such that $\sum_{i,k \in I_g} p'_{i,k}; s_{i,k}; \widetilde{V}_{i,k}; C_g[M, \widetilde{V}_{i,k}] \mathrel{|}\!\!\Longrightarrow_{n'_g} \mathbf{W}_g$ and $\sum_g n'_g = n_2$ and

$$\begin{aligned}&(\mathbf{W}, [\![\textstyle\sum_{j \in J} q_j; t_j; \widetilde{W_j}; C_1[N, \widetilde{W_j}]C_2[N, \widetilde{W_j}]]\!])\\ &\leq_{\mathtt{lift}} \leq_{\mathtt{env}} \{(\mathbf{W}_g, [\![\textstyle\sum_{j,h \in J_g} q'_{j,h}; t_{j,h}; \widetilde{W}_{j,h}; C_g[N, \widetilde{W}_{j,h}]]\!])\}_g\end{aligned}$$

By the inductive hypothesis on each of the n'_g we derive that

$$\{(\mathbf{W}_g, [\![\textstyle\sum_{j,h \in J_g} q_{j,h}; t_{j,h}; \widetilde{W}_{j,h}; C_g[N, \widetilde{W}_{j,h}]]\!])\}_g \subseteq \mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}^{\mathtt{cce}}_{(M,N)}))$$

from which the result follows by $\mathtt{lift}(\geq_{\mathtt{env}} (\,\mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}))\,)) = \mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}))$.
We now consider the case when no $C_1[M, \widetilde{V_i}]$ contributes to the multi-step reduction to $\mathbf{W}$. If there exist some i such that $C_2[M, \widetilde{V_i}]$ contributes to the multi-step reduction to $\mathbf{W}$, then we can derive the result using the same reasoning as above.

Otherwise, there is a set $I' \subseteq I$ such that $C_1[M, \widetilde{V_i}]$ and $C_2[M, \widetilde{V_i}]$ are values for every $i \in I'$, and

$$\begin{aligned}\textstyle\sum_{i \in I'} p_i; s_i; \widetilde{V_i}; C_1[M, \widetilde{V_i}]C_2[M, \widetilde{V_i}] &= \textstyle\sum_{i \in I'} p_i; s_i; \widetilde{V_i}; (\lambda x.P_i)C_2[M, \widetilde{V_i}] \\ &\Longrightarrow_{n_1} \textstyle\sum_{i \in I'} p_i; s_i; \widetilde{V_i}; P_i\{C_2[M, \widetilde{V_i}]/x\} \\ &\longmapsto\!\!\!\Rightarrow_{n_2} \mathbf{W}\end{aligned}$$

with $n_1 \leq n + 1$ and $n_2 \leq n$.
Suppose that for all j we have that $C_1[N, \widetilde{W_j}] = \lambda x.Q_j$ is a value and $C_2[N, \widetilde{W_j}]$ is a value. Then,

$$\begin{aligned}&[\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; C_1[N, \widetilde{W_j}]C_2[N, \widetilde{W_j}]]\!] = \\ &[\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; \lambda x.Q_j C_2[N, \widetilde{W_j}]]\!] = \\ &[\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; Q_j\{C_2[N, \widetilde{W_j}]/x\}]\!]\end{aligned}$$

Since $\mathcal{R}$ is saturated by approximants,

$$\textstyle\sum_{i \in I'} p_i; s_i; \widetilde{V_i} \ \mathcal{R}^{\mathsf{cce}}_{(M,N)} \ \sum_j q_j; t_j; \widetilde{W_j}$$

which implies by Lemma 8.60 that one of the following holds:

- there is a context C' such that

  $$\textstyle\sum_{i \in I'} p_i; s_i; \widetilde{V_i}; P_i\{C_2[M, \widetilde{V_i}]/x\} = \sum_{i \in I'} p_i; s_i; \widetilde{V_i}; C'[M, \widetilde{V_i}]$$

  and

  $$\textstyle\sum_j q_j; t_j; \widetilde{W_j}; Q_j\{C_2[N, \widetilde{W_j}]/x\} = \sum_j q_j; t_j; \widetilde{W_j}; C'[N, \widetilde{W_j}].$$

  Then we can apply the inductive hypothesis on the reduction

  $$\textstyle\sum_{i \in I'} p_i; s_i; \widetilde{V_i}; C'[M, \widetilde{V_i}] \longmapsto\!\!\!\Rightarrow_{n_2} \mathbf{W}$$

  and derive the result.
- for any $\mathbf{W}'$, if $\sum_{i \in I'} p_i; s_i; \widetilde{V_i}, \lambda x.P_i; P_i\{C_2[M, \widetilde{V_i}]/x\} \longmapsto\!\!\!\Rightarrow \mathbf{W}'$ then

  $$\mathbf{W}' \, \mathtt{lift}(\mathcal{R}^{\mathsf{cce}}_{(M,N)}) \, [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}, \lambda x.Q_j; Q_j\{C_2[N, \widetilde{W_j}]/x\}]\!].$$

  Then we have

  $$\begin{aligned}&(\mathbf{W}, [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; Q_j\{C_2[N, \widetilde{W_j}]/x\}]\!]) \\ &\leq_{\mathsf{env}} (\mathbf{W}', [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}, \lambda x.Q_j; Q_j\{C_2[N, \widetilde{W_j}]/x\}]\!])\end{aligned}$$

  and the result follows by $\geq_{\mathsf{env}} \mathtt{lift}(\mathcal{R}) \subseteq \mathtt{lift}(\geq_{\mathsf{env}} (\mathcal{R}))$.

It remains to consider the case when $C_1[N, \widetilde{W_j}]$ or $C_2[N, \widetilde{W_j}]$ are not values for some j. If $C_1[N, \widetilde{W_j}]$ is not a value for some j then $C_1 = [\cdot]_1$ and N is not a value. Suppose $C_2[N, \widetilde{W_j}]$ is a value for all j. Then we have

$$\begin{aligned}&[\![\textstyle\sum_{j \in J} q_j; t_j; \widetilde{W_j}; N C_2[N, \widetilde{W_j}]]\!] = \\ &[\![\textstyle\sum_{j \in J' \subseteq J, h} q_{j,h}; t_{j,h}; \widetilde{W_j}; \lambda x.Q_{j,h} C_2[N, \widetilde{W_j}]]\!] = \\ &[\![\textstyle\sum_{j \in J' \subseteq J, h} q_{j,h}; t_{j,h}; \widetilde{W_j}; Q_{j,h}\{C_2[N, \widetilde{W_j}]/x\}]\!]\end{aligned}$$

for $\llbracket \sum_{j\in J} q_j; t_j; \widetilde{W}_j; N \rrbracket = \sum_{j\in J',h} q_{j,h}; t_{j,h}; \widetilde{W}_j, \lambda x.Q_{j,h}$. Since $\mathcal{R}$ is a finite-step simulation saturated by approximants, by clause (2g) we derive

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V}'_i, \lambda x.P_i \,\mathtt{lift}(\mathcal{R}_{(M,N)})\, \sum_{j\in J',h} q_{j,h}; t_{j,h}; \widetilde{W}'_j, \lambda x.Q_{j,h}$$

which implies

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V}_i, \lambda x.P_i \,\mathtt{lift}(\mathcal{R}^{\mathtt{cce}}_{(M,N)})\, \sum_{j\in J',h} q_{j,h}; t_{j,h}; \widetilde{W}_j, \lambda x.Q_{j,h}\,.$$

Then

$$(\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V}_i, \lambda x.P_i, \sum_{j\in J',h} q_{j,h}; t_{j,h}; \widetilde{W}_j, \lambda x.Q_{j,h}) \leq_{\mathtt{lift}} \{(\mathbf{Y}_g, \mathbf{Z}_g)\}_g \subseteq \mathcal{R}^{\mathtt{cce}}_{(M,N)}$$

with $\mathbf{Y}_g = \sum_{i\in I_g} p'_i; s_i; \widetilde{V}_i, \lambda x.P_i$, and $\mathbf{Z}_g = \sum_{j,h\in J_g} q'_{j,h}; t_{j,h}; \widetilde{W}_j, \lambda x.Q_{j,h}$ such that for every g we have that

$$\textstyle\sum_{i\in I_g} p'_i; s_i; \widetilde{V}_i, \lambda x.P_i; \lambda x.P_i C_2[M, \widetilde{V}_i]$$

and

$$\textstyle\sum_{j,h\in J_g} q'_{j,h}; t_{j,h}; \widetilde{W}_j, \lambda x.Q_{j,h}; \lambda x.Q_{j,h} C_2[N, \widetilde{W}_j]$$

satisfy the premises of Lemma 8.60. Then we can proceed as in the previous case and derive that $\sum_{i\in I_g} p'_i; s_i; \widetilde{V}_i, \lambda x.P_i; P_i\{C_2[M, \widetilde{V}_i]/x\} \Longmapsto \mathbf{W}_g$ implies

$$\mathbf{W}_g \,\mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}^{\mathtt{cce}}_{(M,N)}))\, \llbracket \textstyle\sum_{j,h\in J_g} q'_{j,h}; t_{j,h}; \widetilde{W}_j, \lambda x.Q_{j,h}; Q_{j,h}\{C_2[N, \widetilde{W}_j]/x\} \rrbracket\,,$$

from which the result follows.

Finally, if there are are some j such that $C_2[N, \widetilde{W}_j]$ is not a value then we can proceed as in the previous case, exploiting clause (2g) on $\mathcal{R}_{(M,N)}$ in order to evaluate N in argument position as well.

- case $C = !C'$.
  The interesting case is when for every i we have that $C'[M, \widetilde{V}_i]$ does not contribute to the reduction (otherwise, we can apply the inductive hypothesis analogously to the application case), i.e., there is a subset $I' \subseteq I$ such that

  $$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V}_i; !C'[M, \widetilde{V}_i] = \sum_{i\in I'} p_i; s_i; \widetilde{V}_i; !l_i \Longrightarrow_{n+1} \sum_{i\in I'} p_i; s_i; \widetilde{V}_i, s_i(l_i)$$

  Since $\mathcal{R}$ is saturated by approximants, $\sum_{i\in I'} p_i; s_i; \widetilde{V}'_i \; \mathcal{R}_{(M,N)} \sum_j q_j; t_j; \widetilde{W}'_j$.

  Since contexts are location-free, there are two cases:

  - $C'[M, \widetilde{V}_i] \in \widetilde{V}'_i$. Then $C'[N, \widetilde{W}_j] = l'_j \in \widetilde{W}'_j$ and by clause (2c) on $\mathcal{R}$ we have

    $$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V}'_i, s_i(l_i) \,\mathtt{lift}(\mathcal{R}_{(M,N)})\, \sum_j q_j; t_j; \widetilde{W}'_j, t_i(l'_j)$$

    and the result follows from

    $$\llbracket \textstyle\sum_j q_j; t_j; \widetilde{W}'_j; !l'_j \rrbracket = \sum_j q_j; t_j; \widetilde{W}'_j, t_i(l'_j)$$

- $C = [\cdot]_1$ and $M = l$.
  Suppose that $N = l'$. Then by clause (2g) we have

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i'}, M\ \mathcal{R}_{(M,N)} \sum_j q_j; t_j; \widetilde{W_j'}, N$$

  and by clause (2c)

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i'}, M, s_i(l_i)\ \texttt{lift}(\mathcal{R}_{(M,N)}) \sum_j q_j; t_j; \widetilde{W_j'}, N, t_j(l_j')$$

  Then

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i}, s_i(l_i)\ \texttt{lift}(\geq_{\texttt{env}} (\mathcal{R}^{\texttt{cce}}_{(M,N)})) \sum_j q_j; t_j; \widetilde{W_j}, t_j(l_j')$$

  The case when N is not a value follows analogously.

- $C = C_1 := C_2$.
  Again, the interesting case is when for every i neither $C_1[M, \widetilde{V_i}]$ nor $C_2[M, \widetilde{V_i}]$ contributes to the reduction, i.e., there is a subset $I \subseteq I'$ such that

$$\begin{aligned}&\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i}; C_1[M, \widetilde{V_i}] := C_2[M, \widetilde{V_i}]\\ &= \textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i}; l_i := T_i\\ &\Longmapsto_{n+1} \textstyle\sum_{i\in I'} p_i; s_i[l_i \to T_i]; \widetilde{V_i}, \star\end{aligned}$$

As above, since $\mathcal{R}$ is saturated by approximants, $\sum_{i\in I'} p_i; s_i; \widetilde{V_i'}\ \mathcal{R}_{(M,N)} \sum_j q_j; t_j; \widetilde{W_j'}$ and, since contexts are location-free, there are two cases:

- $C_1[M, \widetilde{V_i}] \in \widetilde{V_i'}$. Then $C_1[N, \widetilde{W_j}] = l_j' \in \widetilde{W_j'}$. Suppose that $C_2[N, \widetilde{W_j}] = U_j$ is a value for every j. Then $(\{T_i\}_i, \{U_j\}_j) \in (\{M, \widetilde{V_i'}\}_i, \{N, \widetilde{W_j'}\}_j)^\star$ and by clause (2c) on $\mathcal{R}$ we have

$$\textstyle\sum_{i\in I'} p_i; s_i[l_i \to T_i]; \widetilde{V_i'}\ \texttt{lift}(\mathcal{R}_{(M,N)}) \sum_j q_j; t_j[l_j' \to U_j]; \widetilde{W_j'}$$

  and the result follows.
  If $C_2[N, \widetilde{W_j}]$ is not a value for some j then $C_2[M, \widetilde{V_j}] = M$ and $C_2[N, \widetilde{W_j}] = N$. Then we derive from clause (2g) that

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i'}, M\ \mathcal{R}_{(M,N)} \llbracket \sum_j q_j; t_j; \widetilde{W_j'}; N\rrbracket = \sum_{j,h} q_{j,h}; t_{j,h}; \widetilde{W_j'}; U_{j,h}$$

  and we can derive, as in the previous case, that

$$\textstyle\sum_{i\in I'} p_i; s_i[l_i \to M]; \widetilde{V_i'}, M\ \texttt{lift}(\mathcal{R}_{(M,N)}) \sum_{j,h} q_j; t_j[l_j' \to U_{j,h}]; \widetilde{W_j'}, U_{j,h}$$

  from which the result follows.

- $C = [\cdot]_1$ and $M = l$.
  Suppose that $N = l'$ and $C_2[N, \widetilde{W_j}] = U_j$ is a value for every j. Then $(\{T_i\}_i, \{U_j\}_j) \in (\{M, \widetilde{V_i'}\}_i, \{N, \widetilde{W_j'}\}_j)^\star$. Then by clause (2g) we have

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i'}, M\ \mathcal{R}_{(M,N)} \sum_j q_j; t_j; \widetilde{W_j'}, N$$

and by clause (2c) on $\mathcal{R}$ we have

$$\textstyle\sum_{i\in I'} p_i; s_i[l \to T_i]; \widetilde{V_i'}, M\ \mathtt{lift}(\mathcal{R}_{(M,N)})\ \sum_j q_j; t_j[l' \to U_j]; \widetilde{W_j'}, N$$

and the result follows.
If N or $C_2[N,\widetilde{W_j}]$ are not values then we proceed analogously to the previous cases, by proving that we can add to the environment the values they evaluate to while staying in relation $\mathcal{R}_{(M,N)}$.

- $C = (\boldsymbol{\nu}\, x := C_1)C_2$, with C_2 a context with free variable x.
We consider the case when

$$\begin{aligned}
&\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i}; (\boldsymbol{\nu}\, x := C_1[M,\widetilde{V_i}])C_2[M,\widetilde{V_i}]\\
&= \textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i}; (\boldsymbol{\nu}\, x := T_i)C_2[M,\widetilde{V_i}]\\
&\Longrightarrow_{n_1} \textstyle\sum_{i\in I'} p_i; s_i[l_i \to T_i]; \widetilde{V_i}; C_2[M,\widetilde{V_i}]\{{}^{l_i}\!/x\}\\
&\Longmapsto_{n_2} \mathbf{W}
\end{aligned}$$

and $C_1[M,\widetilde{V_i}] = U_j$ is a value for every j, thus

$$\begin{aligned}
&[\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; (\boldsymbol{\nu}\, x := C_1[N,\widetilde{W_j}])C_2[N,\widetilde{W_j}]]\!]\\
&= [\![\textstyle\sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}; C_2[N,\widetilde{W_j}]\{{}^{k_j}\!/x\}]\!]
\end{aligned}$$

for $(\{T_i\}_i, \{U_j\}_j) \in (\{M, \widetilde{V_i'}\}_i, \{N, \widetilde{W_j'}\}_j)^{\hat{\star}}$ and for locations $(\{l_i\}_i, \{k_j\}_j)$ which are $(\{s_i\}_i, \{t_j\}_j)$-fresh. By clauses (2d) and (2c) we have

$$\textstyle\sum_{i\in I'} p_i; s_i[l_i \to T_i]; \widetilde{V_i'}, l_i\ \mathcal{R}_{(M,N)}\ \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j'}, k_j$$

which implies that

$$\begin{aligned}
&\textstyle\sum_{i\in I'} p_i; s_i[l_i \to T_i]; \widetilde{V_i}, l_i; C_2[M,\widetilde{V_i}]\{{}^{l_i}\!/x\}\\
&\mathcal{R}^{\mathsf{cce}}_{(M,N)}\ \textstyle\sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}, k_j; C_2[N,\widetilde{W_j}]\{{}^{k_j}\!/x\}\,.
\end{aligned}$$

Then the result follows from the inductive hypothesis on n_2.

- case $C =$ $\mathtt{if}$ C_1 $\mathtt{then}$ C_2 $\mathtt{else}$ C_3 and case $C = \mathtt{op}(C_1, ..., C_m)$.
The result follows from the fact that $\mathcal{R}^{\mathsf{cce}}_{(M,N)}$ satisfies clause (2e) for constants and then from the inductive hypothesis.

- $C = (C_1, ..., C_m)$.
Since the multi-step reduction to $\mathbf{W}$ has length strictly greater than one, there is some z such that $1 \leq z \leq m$ and some i such that $C_z[M,\widetilde{V_i}]$ contributes to the multi-step reduction to $\mathbf{W}$. Then we can apply the inductive hypothesis on the contexts to C_z and derive that:

$$\textstyle\sum_{i\in I'} p_i; s_i; \widetilde{V_i}; C_z[M,\widetilde{V_i}] \Longmapsto_{n'} \sum_{i\in I',k} p_{i,k}; s_{i,k}; \widetilde{V_i}, V_{i,k}$$

for $n' \leq n+1$ implies

$$\begin{aligned}
\textstyle\sum_{i\in I',k} p_{i,k}; s_{i,k}; \widetilde{V_i}, V_{i,k}\ \mathtt{lift}(\geq_{\mathtt{env}} (\mathcal{R}^{\mathsf{cce}}_{(M,N)}))\ &\textstyle\sum_{j\in J',h} q_{j,h}; t_{j,h}; \widetilde{W_j}, W_{j,h}\\
&= [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}; C_z[N,\widetilde{W_j}]]\!]
\end{aligned}$$

i.e.,

$$(\textstyle\sum_{i\in I',k} p_{i,k}; s_{i,k}; \widetilde{V_i}, V_{i,k}, \sum_{j\in J',h} q_{j,h}; t_{j,h}; \widetilde{W_j}, W_{j,h}) \leq_{\mathtt{lift}}\leq_{\mathtt{env}} \{(\mathbf{Y}_g, \mathbf{Z}_g)\}_g \subseteq\mathcal{R}^{\mathtt{cce}}_{(M,N)}$$

with $\mathbf{Y}_g = \sum_{i,k\in I_g} p'_{i,k}; s_{i,k}; \widetilde{V_{i,k}}$, and $\mathbf{Z}_g = \sum_{j,h\in J_g} q'_{j,h}; t_{j,h}; \widetilde{W_{j,h}}$ and for every g there is a context C_g such that for every $i, k \in I_g$ and $j, h \in J_g$:

$$\begin{aligned} C[M, \widetilde{V_i}]\{{}^{V_{i,k}}/C_z[M, \widetilde{V_i}]\} &= C_g[M, \widetilde{V_{i,k}}] \\ C[N, \widetilde{W_j}]\{{}^{W_{j,h}}/C_z[N, \widetilde{W_j}]\} &= C_g[N, \widetilde{W_{j,h}}] \end{aligned}$$

(where the substitution only concerns the specific instance of $C_z[M, \widetilde{V_i}]$ in $C_z[M, \widetilde{V_i}]$ that occurs as the z-th element of the tuple $C[M, \widetilde{V_i}]$, and the same for $C_z[N, \widetilde{W_i}]$). Finally, we derive the result by applying the inductive hypothesis on the number of reductions from

$$\textstyle\sum_{i,k\in I_g} p'_{i,k}; s_{i,k}; \widetilde{V_{i,k}}; C_g[M, \widetilde{V_{i,k}}]\,.$$

- $C = \#_z(C')$.
  We consider the case when $C'[M, \widetilde{V_i}] = (V_{i,1}, ..., V_{i,m})$ is a value for every i and

  $$\textstyle\sum_{i\in I} p_i; s_i; \widetilde{V_i}; \#_z(C'[M, \widetilde{V_i}]) \longrightarrow \mathbf{W} = \sum_{i\in I} p_{i,k}; s_{i,k}; \widetilde{V_i}, V_{i,a}$$

  We have three cases:

  - $C'[M, \widetilde{V_i}] = (V_{i,1}, ..., V_{i,m}) \in \widetilde{V_i'}$.
    Then $C'[N, \widetilde{W_j}] = (U_{j,1}, ..., U_{j,m}) \in \widetilde{W_j'}$ and we have by clause (2f) on $\mathcal{R}$ that

    $$\textstyle\sum_i p_i; s_i; \widetilde{V_i'}, V_{i,1}, ..., V_{i,m}\, \mathtt{lift}(\mathcal{R}_{(M,N)})\, \sum_j q_j; t_j; \widetilde{W_j'}, U_{j,1}, ..., U_{j,m}$$

    Therefore,

    $$\begin{aligned} &(\mathbf{W}, [\![\textstyle\sum_j q_j; t_j; \widetilde{W_j}, \#_z(U_{j,1}, ..., U_{j,m})]\!]) \\ &\leq_{\mathtt{env}} (\textstyle\sum_i p_i; s_i; \widetilde{V_i}, V_{i,1}, ..., V_{i,m}, \sum_j q_j; t_j; \widetilde{W_j}, U_{j,1}, ..., U_{j,m}) \in \mathtt{lift}(\mathcal{R}^{\mathtt{cce}}_{(M,N)}) \end{aligned}$$

  - $C'[M, \widetilde{V_i}] = (C_1[M, \widetilde{V_i}], ..., C_m[M, \widetilde{V_i}])$.
    Then $C'[N, \widetilde{W_j}] = (C_1[N, \widetilde{W_j}], ..., C_m[N, \widetilde{W_j}])$ and the result directly follows from the definition of the preorder $\leq_{\mathtt{cce}(M,N)}$.
  - $C' = [\cdot]_1$.
    The result follows from clauses (2g) and (2f) for $\mathcal{R}$.

□

Lemma 8.62. *Suppose that $\mathcal{R}_{(M,N)}$ is a finite-step $\{\widetilde{l}\}$-simulation saturated by approximants (only defined on formal sums), and that C is a context with locations in $\{\widetilde{l}\}$. The following relation satisfies the clauses on formal sums for finite-step simulations up-to lifting and environment:*

$$\mathcal{S}= \{((C[M], C[N]), \mathbf{Y}, \mathbf{Z}) \mid \mathbf{Y}\ \mathcal{R}^{\mathtt{cce}}_{(M,N)}\ \mathbf{Z}\}$$

Proof. We can assume that the relation $\mathcal{R}_{(M,N)}$ is *closed by* $\{\widetilde{l}\}$, i.e., each location $l \in \{\widetilde{l}\}$ occurs in corresponding columns in the dynamic environment of the formal sums (formally: for any $\mathbf{Y}, \mathbf{Z}$ in the relation and for every $l \in \{\widetilde{l}\}$ there is an index r, for $1 \leq r \leq\mid \mathbf{Y} \mid$, such that both $\mathbf{Y}\lfloor_r$ and $\mathbf{Z}\lfloor_r$ are tuples composed by location l). This assumption simplifies our proof, while not affecting the results. Indeed, we can eliminate all the pairs that do not satisfy the requirement of $\{\widetilde{l}\}$-closure and we still have a finite-step $\{\widetilde{l}\}$-simulation saturated by approximants.

The proof exploits the fact that if $\mathcal{R}_{(M,N)}$ is closed by $\{\widetilde{l}\}$ then $\mathcal{R}^{\mathtt{cce}}_{(M,N)}$ is closed by $\{\widetilde{l}\}$ and, for any C such that $\mathtt{Loc}(C) \subseteq \{\widetilde{l}\}$, if $\sum_i p_i; s_i; \widetilde{V_i}\ \mathcal{R}^{\mathtt{cce}}_{(M,N)}\ \sum_j q_j; t_j; \widetilde{W_j}$ and $(\{T_i\}_i, \{U_j\}_j) \in (\{C[M], \widetilde{V_i}\}_i, \{C[N], \widetilde{W_j}\}_j)^{\star}$ then $(\{T_i\}_i, \{U_j\}_j) \in (\{M, \widetilde{V_i}\}_i, \{N, \widetilde{W_j}\}_j)^{\star}$, since the locations in $\{\widetilde{l}\}$ are guaranteed to occur at corresponding columns in $\{\widetilde{V_i}\}_i$ and $\{\widetilde{W_j}\}_j$ and then C can be turned into a location-free context.

The condition (2a) on the weights is immediately satisfied by the definition of $\mathcal{R}^{\mathtt{cce}}_{(M,N)}$, since $\mathcal{R}$ is a simulation.

Then we prove that the conditions from (2b) to (2g) of finite-step simulation up-to lifting and environment are satisfied by $\mathcal{S}$. Suppose that $\mathbf{Y} = \sum_i p_i; s_i; \widetilde{V_i}\ \mathcal{S}_{(C[M],C[N])}\ \sum_j q_j; t_j; \widetilde{W_j} = \mathbf{Z}$ with $\mathbf{Y}' = \sum_i p_i; s_i; \widetilde{V_i'}, \mathbf{Z}' = \sum_j q_j; t_j; \widetilde{W_j'}$ related by $\mathcal{R}_{(M,N)}$ and such that $(\mathbf{Y}', \mathbf{Z}') \leq_{\mathtt{cce}(M,N)} (\mathbf{Y}, \mathbf{Z})$.

(2b) for all r, if $(\widetilde{V_i})_r = \lambda x.M_i$ and $(\widetilde{W_j})_r = \lambda x.N_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{C[M], \widetilde{V_i}\}_i, \{C[N], \widetilde{W_j}\}_j)^{\star}$, if $\sum_i p_i; s_i; \widetilde{V_i}; M_i\{T_i/x\} \Longmapsto \mathbf{W}$ then

$$\mathbf{W}\,\mathtt{lift}(\geq_{\mathtt{env}}(\mathcal{S}_{(C[M],C[N])}))\ \textstyle\sum_j q_j; \widetilde{W_j} \cdot [\![\langle t_j\ ;\ N_j\{U_j/x\}\rangle]\!]$$

Proof. We have three cases:

- if $(\widetilde{V_i})_r = \lambda x.C'[M, \widetilde{V_i'}]$ and $(\widetilde{W_j})_r = \lambda x.C'[N, \widetilde{W_j'}]$ for some location-free context C' then, since $\mathcal{R}_{(M,N)}$ is closed with respect to $\widetilde{l}$, there is a location-free context C'' such that

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i}; M_i\{T_i/x\} = \sum_i p_i; s_i; \widetilde{V_i}; C''[M, \widetilde{V_i}]$$
$$\textstyle\sum_j q_j; t_j; \widetilde{W_j}; N_j\{U_j/x\} = \sum_j q_j; t_j; \widetilde{W_j}; C''[N, \widetilde{W_j}]$$

  and we derive the result by Lemma 8.61.

- if $(\widetilde{V_i})_r = (\widetilde{V_i'})_{r'}$ and $(\widetilde{W_j})_r = (\widetilde{W_j'})_{r'}$ for some r' then, since $\mathcal{R}_{(M,N)}$ is closed with respect to $\widetilde{l}$, there is a location-free context C'' such that

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i}; M_i\{T_i/x\} = \sum_i p_i; s_i; \widetilde{V_i}; M_i\{C''[M, \widetilde{V_i'}]/x\}$$
$$\textstyle\sum_j q_j; t_j; \widetilde{W_j}; N_j\{U_j/x\} = \sum_j q_j; t_j; \widetilde{W_j}; N_j\{C''[N, \widetilde{W_j'}]/x\}$$

  Since $\mathcal{R}_{(M,N)}$ is a finite-step simulation,

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i'}; M_i\{C''[M, \widetilde{V_i'}]/x\} \Longmapsto \mathbf{W}$$

  implies $\mathbf{W}\,\mathtt{lift}(\mathcal{R}_{M,N})\,[\![\sum_j q_j; t_j; \widetilde{W_j'}; N_j\{C''[N, \widetilde{W_j'}]/x\}]\!]$, which in turn implies the result analogously to Lemma 8.60.

- if $(\widetilde{V_i})_r = M$ and $(\widetilde{W_j})_r = N$ then M and N are values and by clause (2g) applied to $\mathcal{R}_{(M,N)}$ we have

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i'}, M \,\texttt{lift}(\mathcal{R}_{(M,N)})\, \sum_j q_j; t_j; \widetilde{W_j'}, N$$

Hence, by clause (2b) applied to $\mathcal{R}_{(M,N)}$ we have

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i'}, M; M_i\{{}^{C''[M,\widetilde{V_i'}]}\!/x\} \Longmapsto \mathbf{W}$$

implies $\mathbf{W}\,\texttt{lift}(\mathcal{R}_{(M,N)})\, [\![\sum_j q_j; t_j; \widetilde{W_j'}, N; N_j\{{}^{U_j}\!/x\}]\!]$, which in turn implies the result (see the proof of Lemma 8.60).

□

(2c) for all r, if $(\widetilde{V_i})_r = l_i$ and $(\widetilde{W_j})_r = k_j$ then

- $\sum_i p_i; s_i; \widetilde{V_i}, s_i(l_i)\, \texttt{lift}(\mathcal{S}_{(C[M],C[N])})\, \sum_j q_j; t_j; \widetilde{W_j}, t_j(k_j)$,
- for all $(\{T_i\}_i, \{U_j\}_j) \in (\{C[M], \widetilde{V_i}\}_i, \{C[N], \widetilde{W_j}\}_j)^{\widehat{\star}}$ we have

$$\textstyle\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i}\; \mathcal{S}_{(C[M],C[N])}\; \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}\,.$$

Proof. Since contexts used in $\leq_{\texttt{cce}(M,N)}$ are location-free and $\mathcal{R}_{(M,N)}$ is closed with respect to the locations in $\widetilde{l}$ (and thereby the locations in $C[M], C[N]$ are in the dynamic environment of the formal sums in $\mathcal{R}_{(M,N)}$ in corresponding columns), there is an r' such that $(\widetilde{V_i'})_{r'} = l_i$ and $(\widetilde{W_j'})_{r'} = k_j$, then:

- $\sum_i p_i; s_i; \widetilde{V_i'}, s_i(l_i)\, \texttt{lift}(\mathcal{R}_{(M,N)})\, \sum_j q_j; t_j; \widetilde{W_j'}, t_j(k_j)$ and we have

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i}, s_i(l_i)_i\, \texttt{lift}(\mathcal{R}^{\texttt{cce}}_{(M,N)})\, \sum_j q_j; t_j; \widetilde{W_j}, t_j(k_j)_j,$$

from which the result follows.
- by $(\{T_i\}_i, \{U_j\}_j) \in (\{M, \widetilde{V_i'}\}_i, \{N, \widetilde{W_j'}\}_j)^{\widehat{\star}}$ we derive

$$\textstyle\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i'}\; \mathcal{R}_{(M,N)}\; \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j'}\,,$$

which implies

$$\textstyle\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i}\; \mathcal{S}_{(C[M],C[N])}\; \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}\,.$$

□

(2d) for any $(\{s_i\}_i, \{t_j\}_j)$-fresh locations $(\{l_i\}_i, \{k_j\}_j)$,
and for all $(\{T_i\}_i, \{U_j\}_j) \in (\{C[M], \widetilde{V_i}\}_i, \{C[N], \widetilde{W_j}\}_j)^{\widehat{\star}}$,

$$\textstyle\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V_i}, l_i\; \mathcal{S}_{(C[M],C[N])}\; \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W_j}, k_j\,.$$

Proof. The result follows from $(\{T_i\}_i, \{U_j\}_j) \in (\{M, \widetilde{V_i'}\}_i, \{N, \widetilde{W_j'}\}_j)^{\widehat{\star}}$ as in the previous clause, by exploiting clause (2d) on $\mathcal{R}$. □

(2e) for all r, if $(\widetilde{V_i})_r = c_i$ and $(\widetilde{W_j})_r = c_j$ then all constants in the two columns are the same (i.e., there is c_a with $c_i = c_j = c_a$ for all i, j).

Proof. For every r, there are three cases: either $\mathbf{Y}\lfloor_r = \mathbf{Y}'\lfloor_{r'}$ and $\mathbf{Z}\lfloor_r = \mathbf{Z}'\lfloor_{r'}$ for some r', in which case both columns are composed by the same constant, since $\mathcal{R}$ is a finite-step simulation; or the value-context is a constant and $\mathbf{Y}\lfloor_r = \mathbf{Z}\lfloor_r = c$; or the constants are respectively M and N, in which case

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i'}, M \,\mathtt{lift}(\mathcal{R}_{(M,N)})\, \sum_j q_j; t_j; \widetilde{W_j'}, N$$

(by clause (2g)) and thus they have to be the same constant, otherwise $\mathcal{R}$ would not respect condition (2e). □

(2f) for all r, if $(\widetilde{V_i})_r = (V_{i,1}, ..., V_{i,n})$ and $(\widetilde{W_j})_r = (W_{j,1}, ..., W_{j,n})$ then

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i}, V_{i,1}, ..., V_{i,n} \,\mathtt{lift}(\mathcal{S}_{(C[M],C[N])})\, \sum_j q_j; t_j; \widetilde{W_j}, W_{j,1}, ..., W_{j,n} \,.$$

Proof. If $\mathbf{Y}\lfloor_r = \mathbf{Y}'\lfloor_{r'}$ and $\mathbf{Z}\lfloor_r = \mathbf{Z}'\lfloor_{r'}$ for some r' then it follows from the definition of $\mathcal{R}$ that

$$\textstyle\sum_i p_i; s_i; \widetilde{V_i}, V_{i,1}, ..., V_{i,n} \,\mathtt{lift}(\mathcal{S}_{(C[M],C[N])})\, \sum_j q_j; t_j; \widetilde{W_j}, W_{j,1}, ..., W_{j,n}.$$

Otherwise, for $1 \leq h \leq n$ we have that $(\{V_{i,h}\}_i, \{U_{j,h}\}_j) \in (\{M, \widetilde{V_i'}\}_i, \{N, \widetilde{W_j'}\}_j)^{\widehat{\star}}$, which implies that

$$(\mathbf{Y}', \mathbf{Z}') \leq_{\mathsf{cce}(M,N)} (\textstyle\sum_i p_i; s_i; \widetilde{V_i}, V_{i,1}, ..., V_{i,n}, \sum_j q_j; t_j; \widetilde{W_j}, W_{j,1}, ..., W_{j,n}),$$

i.e., the formal sums are in relation $\mathtt{lift}(\mathcal{S}_{(C[M],C[N])})$.
Finally, if $(\widetilde{V_i})_r = M$ and $(\widetilde{W_j})_r = N$ then the clause follows as in the previous cases from $\sum_i p_i; s_i; \widetilde{V_i'}, M \,\mathtt{lift}(\mathcal{R}_{(M,N)})\, \sum_j q_j; t_j; \widetilde{W_j'}, N$, by clause (2g). □

(2g) $\sum_i p_i; \widetilde{V_i} \cdot [\![\langle s_i \,;\, C[M]\rangle]\!] \,\mathtt{lift}(\mathcal{S}_{(C[M],C[N])})\, \sum_j q_j; \widetilde{W_j} \cdot [\![\langle t_j \,;\, C[N]\rangle]\!]$.

Proof. Since $\mathcal{R}_{(M,N)}$ is closed with respect to $\widetilde{l}$, there is a location-free context C' such that

$$\begin{array}{c}\mathbf{Y}; C[M] = \mathbf{Y}; C'[M, \mathbf{Y}] \\ \mathbf{Z}; C[N] = \mathbf{Z}; C'[N, \mathbf{Z}]\end{array}$$

and the result follows from Lemma 8.61. □

□

We can now derive from Lemma 8.62 the congruence result. Let C be a context such that $\mathtt{Loc}(C) \subseteq \{\widetilde{l}\}$. Let $\mathcal{R}$ be a finite-step $\{\widetilde{l}\}$-simulation such that $\langle s \,;\, M\rangle \;\mathcal{R}\; \langle t \,;\, N\rangle$. Then the relation

$$\{(\langle s \,;\, C[M]\rangle, \langle t \,;\, C[N]\rangle)\} \cup \{((C[M], C[N]), \mathbf{Y}, \mathbf{Z}) \mid \mathbf{Y} \;\mathcal{R}^{\mathsf{cce}}_{(M,N)}\; \mathbf{Y}\}$$

is a finite step simulation up-to lifting and environment, since:

- clause (1) on terms follows since $\mathtt{Loc}(C) \subseteq \{\widetilde{l}\}$ and by clause (1) for $\mathcal{R}$ we have that $(1; s; \widetilde{l}, 1; t; \widetilde{l}) \in \mathcal{R}_{(M,N)} \subseteq \mathcal{R}^{\mathsf{cce}}_{(M,N)}$;
- the clauses for formal sums follow by Lemma 8.62.

Proof of Theorem 8.52

Let $\widetilde{l'} = \widetilde{l}, \widetilde{l''}$ and let $\widetilde{V'} = \widetilde{V}, \widetilde{V''}$ be a sequence of values whose types are consistent with those of $\widetilde{l'}$ and with locations in $\{\widetilde{l'}\}$. Let C be a context with locations in $\{\widetilde{l'}\}$ and let $\mathcal{R}$ be a finite-step $\{\widetilde{l}\}$-simulation (saturated by approximants) relating $\langle s\,;\, M\rangle$ and $\langle t\,;\, N\rangle$. Then $1; s; \widetilde{l}\ \mathcal{R}_{(M,N)}\ 1; t; \widetilde{l}$ and by repeatedly applying clause (2d) we derive that

$$1; s[\widetilde{l''} \to \widetilde{W}]; \widetilde{l'}\ \mathcal{R}_{(M,N)}\ 1; t[\widetilde{l''} \to \widetilde{W}]; \widetilde{l'}$$

for a consistent sequence of values $\widetilde{W}$. (Note that we cannot guarantee by just using clause (2d) that the tuple of values $\widetilde{V''}$ is assigned to $\widetilde{l''}$, since locations in $\widetilde{l''}$ might occur in any value in $\widetilde{V''}$. Hence, we first have to put all the locations in $\{\widetilde{l''}\}$ in the dynamic environment.) Then by repeatedly applying clause (2c) we derive

$$1; s[\widetilde{l''} \to \widetilde{V''}]; \widetilde{l'}\ \mathcal{R}_{(M,N)}\ 1; t[\widetilde{l''} \to \widetilde{V''}]; \widetilde{l'}\,.$$

It is easy to see that if we restrict $\mathcal{R}_{(M,N)}$ to those pairs of formal sums whose dynamic environments begin with the sequence $\widetilde{l'}$ of locations then the clauses of finite-step $\{\widetilde{l'}\}$-simulation are satisfied. Let $\mathcal{R}'_{(M,N)}$ be such a restriction of $\mathcal{R}_{(M,N)}$. Then we can apply Lemma 8.62 (see the proof of Theorem 8.51) and derive that relation

$$\mathcal{S} = \{((C[M], C[N]), \mathbf{Y}, \mathbf{Z})\ \mid\ \mathbf{Y}\ \mathcal{R}'^{\mathsf{cce}}_{(M,N)}\ \mathbf{Z}\}$$

is a finite-step $\{\widetilde{l'}\}$-simulation (up-to lifting and environment). Since

$$(1; s[\widetilde{l''} \to \widetilde{V''}]; \widetilde{l'}, 1; t[\widetilde{l''} \to \widetilde{V''}]; \widetilde{l'}) \in \mathcal{R}'_{(M,N)} \subseteq \mathcal{R}'^{\mathsf{cce}}_{(M,N)} = \mathcal{S}_{(C[M],C[N])}$$

we conclude $\langle s[\widetilde{l''} \to \widetilde{V''}]\,;\, C[M]\rangle \lesssim^{\{\widetilde{l'}\}}_{\text{fin}} \langle t[\widetilde{l''} \to \widetilde{V''}]\,;\, C[N]\rangle$.

Finally, by repeatedly applying clause (2c) to locations $\widetilde{l}$ in the pair

$$1; s[\widetilde{l''} \to \widetilde{V''}]; \widetilde{l'}\ \mathcal{S}_{(C[M],C[N])}\ 1; t[\widetilde{l''} \to \widetilde{V''}]; \widetilde{l'}$$

we derive

$$1; \widetilde{l'} = \widetilde{V'}; \widetilde{l'}\ \mathcal{S}_{(C[M],C[N])}\ 1; \widetilde{l'} = \widetilde{V'}; \widetilde{l'}$$

which in turn implies $\langle \widetilde{l'} = \widetilde{V'}\,;\, C[M]\rangle \lesssim^{\{\widetilde{l'}\}}_{\text{fin}} \langle \widetilde{l'} = \widetilde{V'}\,;\, C[N]\rangle$.

Proof of Theorem 8.54

We prove that the relation

$$\begin{aligned}\mathcal{R}\ = &\{(\langle \widetilde{l} = \widetilde{V}\,;\, M\rangle, \langle \widetilde{l} = \widetilde{V}\,;\, N\rangle)\ \mid\ M \leq_{\mathtt{ctx}} N\ \wedge\ \{\widetilde{l}\} = \mathtt{Loc}(M) \cup \mathtt{Loc}(N)\} \cup\\ &\{((M,N), \textstyle\sum_i p_i; s_i; V^i_1, ..., V^i_n\,,\ \sum_j q_j; t_j; W^j_1, ..., W^j_n)\ \mid\ M \leq_{\mathtt{ctx}} N\\ &\wedge\ \exists C, \widetilde{V} \text{ such that } ([\![\langle \widetilde{l} = \widetilde{V}\,;\, C[M]\rangle]\!] = \textstyle\sum_i p_i; s_i; \lambda x.xV^i_1...V^i_n\\ &\wedge\ [\![\langle \widetilde{l} = \widetilde{V}\,;\, C[N]\rangle]\!] = \textstyle\sum_j q_j; t_j; \lambda x.xW^j_1...W^j_n\\ &\text{with } \mathtt{Loc}(C) \subseteq \{\widetilde{l}\} = \mathtt{Loc}(M) \cup \mathtt{Loc}(N)\\ &\wedge\ \text{they are } \textit{first-order consistent})\}\end{aligned}$$

satisfies the clauses of $\{\widetilde{l}\}$-simulation. Let $\widetilde{l} = l_1, ...l_n = \texttt{Loc}(M) \cup \texttt{Loc}(N)$ and $\langle \widetilde{l} = \widetilde{V}\,;\, M\rangle \;\mathcal{R}\; \langle \widetilde{l} = \widetilde{V}\,;\, N\rangle$. Hence, $M \leq_{\texttt{ctx}} N$ and clause (1) holds since, using context $C = \lambda x.xl_1...l_n$ (with no holes) we derive from $M \leq_{\texttt{ctx}} N$ that $1; \widetilde{l} = \widetilde{V}; l_1, ..., l_n \;\mathcal{R}_{(M,N)}\; 1; \widetilde{l} = \widetilde{V}; l_1, ..., l_n$.

To prove that $\mathcal{R}$ satisfies the clauses of simulation for formal sums, we first show the following lemma.

Lemma 8.63. *If* $\mathbf{Y} \;\mathcal{R}_{(M,N)}\; \mathbf{Z}$ *then for any* C *with* $\texttt{Loc}(C) \subseteq \texttt{Loc}(M) \cup \texttt{Loc}(N)$*:*

- *If* $[\![\mathbf{Y}; C[M,\mathbf{Y}]]\!]$ *and* $[\![\mathbf{Z}; C[N,\mathbf{Z}]]\!]$ *are first-order consistent then*
$$[\![\mathbf{Y}; C[M,\mathbf{Y}]]\!] \;\mathcal{R}_{(M,N)}\; [\![\mathbf{Z}; C[N,\mathbf{Z}]]\!]\,;$$
- *If* $[\![\mathbf{Y}; C[M,\mathbf{Y}]]\!]$ *and* $[\![\mathbf{Z}; C[N,\mathbf{Z}]]\!]$ *are not first-order consistent then*
$$[\![\mathbf{Y}; C[M,\mathbf{Y}]]\!] \;\texttt{lift}(\mathcal{R}_{(M,N)})\; [\![\mathbf{Z}; C[N,\mathbf{Z}]]\!]\,.$$

Proof. If $\mathbf{Y} = \sum_i p_i; s_i; V^i_1, ..., V^i_n \;\mathcal{R}_{(M,N)}\; \sum_j q_j; t_j; W^j_1, ..., W^j_n = \mathbf{Z}$ then they are first-order consistent environment formal sums and there are C, s such that $[\![\langle s\,;\, C[M]\rangle]\!] = \sum_i p_i; s_i; \lambda x.xV^i_1...V^i_n$ and $[\![\langle s\,;\, C[N]\rangle]\!] = \sum_j q_j; t_j; \lambda x.xW^j_1...W^j_n$ and $\texttt{Loc}(C) \subseteq \{\widetilde{l}\} = \texttt{Loc}(M) \cup \texttt{Loc}(N)$.

Let C' be any context with $\texttt{Loc}(C) \subseteq \{\widetilde{l}\} = \texttt{Loc}(M) \cup \texttt{Loc}(N)$ and let

$$P_{M,C'} = \lambda x_1, ..., x_n.(\lambda z, x.xx_1...x_nz)C'[M, x_1, ..., x_n]$$

and $P_{N,C'}$ the same term with M substituted to N. It follows from $M \leq_{\texttt{ctx}} N$ that $C[M]P_{M,C'} \leq_{\texttt{ctx}} C[N]P_{N,C'}$.

We have that:

$$\begin{aligned}
[\![\langle s\,;\, C[M]P_{M,C'}\rangle]\!] &= [\![[\![\langle s\,;\, C[M]\rangle]\!]P_{M,C'}]\!] \\
&= [\![\textstyle\sum_i p_i; s_i; (P_{M,C'}V^i_1...V^i_n)]\!] \\
&= [\![\textstyle\sum_i p_i; s_i; (\lambda z, x.xV^i_1...V^i_nz)C'[M, V^i_1, ..., V^i_n]]\!] \\
&= \textstyle\sum_{i,k} p_{i,k}; s_{i,k}; (\lambda z, x.xV^i_1...V^i_nV_{i,k})
\end{aligned}$$

for $[\![\sum_i p_i; s_i; C'[M, V^i_1, ..., V^i_n]]\!] = \sum_{i,k} p_{i,k}; s_{i,k}; V_{i,k}$ and analogously for N:

$$[\![\langle s\,;\, C[N]P_{N,C'}\rangle]\!] = \textstyle\sum_{j,h} q_{j,h}; t_{j,h}; \lambda x.xW^j_1...W^j_nW_{j,h}$$

for $[\![\sum_j q_j; t_j; C'[N, W^j_1, ..., W^j_n]]\!] = \sum_{j,k} q_{j,h}; t_{j,h}; W_{j,h}$.
If $[\![\sum_i p_i; s_i; C'[M, V^i_1, ..., V^i_n]]\!]$ and $[\![\sum_j q_j; t_j; C'[N, W^j_1, ..., W^j_n]]\!]$ are first order consistent, then we can conclude, by the definition of $\mathcal{R}$, that

$$\begin{aligned}
&\textstyle\sum_i p_i; V^i_1, ..., V^i_n \cdot [\![\langle s_i\,;\, C'[M, V^i_1, ..., V^i_n]\rangle]\!] \\
&\mathcal{R}_{(M,N)} \textstyle\sum_j q_j; W^j_1, ..., W^j_n \cdot [\![\langle t_j\,;\, C'[N, W^j_1, ..., W^j_n]\rangle]\!]
\end{aligned}$$

If $[\![\sum_i p_i; s_i; C'[M, V^i_1, ..., V^i_n]]\!]$ and $[\![\sum_j q_j; t_j; C'[N, W^j_1, ..., W^j_n]]\!]$ are not first order consistent, which means that the dynamic environment is composed of different constants, then for any constant c we can use the term

$$P_{M,C',c} = \lambda x_1, ..., x_n.(\lambda z, x.xx_1...x_nz)\;\texttt{if}\;\; C'[M, x_1, ..., x_n] = c\;\;\texttt{then}\;\; c\;\;\texttt{else}\;\;\Omega$$

to derive, analogously as above, that

$$\textstyle\sum_{\{i,k|V_{i,k}=c\}} p_{i,k}; s_{i,k}; V_1^i, ..., V_n^i, V_{i,k}\ \mathcal{R}_{(M,N)}\ \sum_{\{j,h|W_{j,h}=c\}} q_{j,h}; t_{j,h}; W_1^j, ..., W_n^j, W_{j,h}$$

and thus

$$\begin{aligned}\mathbf{Y}; C[M, \mathbf{Y}] = &\textstyle\sum_c \sum_{\{i,k|V_{i,k}=c\}} p_{i,k}; s_{i,k}; V_1^i, ..., V_n^i, V_{i,k}\\ &\mathtt{lift}(\mathcal{R}_{(M,N)})\\ &\textstyle\sum_c \sum_{\{j,h|W_{j,h}=c\}} q_{j,h}; t_{j,h}; W_1^j, ..., W_n^j, W_{j,h} = \mathbf{Z}; C[N, \mathbf{Z}]\end{aligned}$$

□

Let $\mathbf{Y} = \sum_i p_i; s_i; V_1^i, ..., V_n^i\ \mathcal{R}_{(M,N)}\ \sum_j q_j; t_j; W_1^j, ..., W_n^j) = \mathbf{Z}$ be first-order consistent environment formal sums and let C, s be such that $[\![\langle s\,;\, C[M]\rangle]\!] = \sum_i p_i; s_i; \lambda x.xV_1^i...V_n^i$ and $[\![\langle s\,;\, C[N]\rangle]\!] = \sum_j q_j; t_j; \lambda x.xW_1^j...W_n^j$. We can now prove that $\mathcal{R}$ satisfies the simulation clauses on formal sums.

(2a) It follows from the definition of $\leq_{\mathtt{ctx}}$ that $M \leq_{\mathtt{ctx}} N$ implies $C[M] \leq_{\mathtt{ctx}} C[N]$, which implies $\mathtt{weight}(\mathbf{Y}) = \mathtt{weight}([\![\langle s\,;\, C[M]\rangle]\!]) \leq \mathtt{weight}([\![\langle s\,;\, C[N]\rangle]\!]) = \mathtt{weight}(\mathbf{Z})$.

(2b) Let $V_r^i = \lambda x.M_i$ and $W_r^j = \lambda x.N_j$. The result follows from Lemma 8.63, using context $C = [\cdot]_{r+1}C'$, for any value context C'.

(2c) Let $V_r^i = l_i$ and $W_r^j = k_j$. The result follows from Lemma 8.63, respectively using contexts $C_1 = ![\cdot]_{r+1}$ and $C_2 = [\cdot]_{r+1} := C'$, for any value context C'. In the latter case, since the formal sums are first-order consistent we can use directly relation $\mathcal{R}_{(M,N)}$, without the lifting construction (by the first item of Lemma 8.63).

(2d) The result follows from the first item of Lemma 8.63, using context $C = (\boldsymbol{\nu}\, x{:=}C_1)C_2$, for C_1 a value context and C_2 a context with free variable x.

(2e) The result directly follows from the definition of $\mathcal{R}$.

(2f) For all r, if $V_r^i = (V_{i,1}, ..., V_{i,n})$ and $W_r^j = (W_{j,1}, ..., W_{j,n})$ then the result follows by iteratively applying Lemma 8.63, using contexts $C_1 = \#_1([\cdot]_{r+1})$,..., $C_n = \#_n([\cdot]_{r+1})$.

(2g) The result follows from Lemma 8.63, using context $C = [\cdot]_1$.

Chapter 9

Conclusions

9.1 Additional related works

We discuss here some additional works on probabilistic calculi based on different notions of bisimulations (with respect to applicative or environmental) or different techniques for proving contextual equivalence.

In [DLSA14] and [CD14] probabilistic applicative bisimulations for pure call-by-name and call-by-value λ-calculi are shown to be congruences. Completeness however only holds in call-by-value, while it fails in call-by-name. In call-by-name, completeness is obtained using *coupled logical bisimulation*, a probabilistic version of the logical bisimilarity for deterministic languages [SKS07]. While applicative bisimulation requires two functions to be related whenever they take as input the same argument, logical bisimulation requires two functions to be related whenever they take as input terms in the contextual closure of the relation itself. Since the contextual closure of a relation includes identity, the set of terms with which related functions are tested is enlarged with respect to applicative bisimilarity. This makes the congruence proof easier by allowing a direct use of the inductive hypothesis, thereby removing the need for Howe's technique. Drawbacks of all forms of logical bisimilarity are a non-monotone functional (which makes it harder to prove that bisimilarity is the largest bisimulation) and a confinement to pure λ-calculi. Further, up-to techniques may be difficult in logical bisimilarity. For instance, Example 8.25 cannot be proved with the techniques in [DLSA14]: the equality fails for applicative bisimilarity, and the up-to context technique provided for logical bisimilarity is not powerful enough (the paper shows a similar example, akin to Example 8.5, where however the functions employed immediately throw away their input, and this is essential for the proof).

An alternative bisimulation for enriched calculi is *normal form* (or *open*) bisimulation [San94; SL07; LL07; JPR09]. This is complete (with respect to contextual equivalence) only in certain extensions of the λ-calculus (e.g., call-by-value with *both* state *and* callcc), and would be incomplete in other languages (such as λ-calculus without state or/and callcc, and languages with constants or types).

Another approach to contextual equivalence in higher-order languages is via logical relations (see, e.g., [Mit96, Chapter 8] and [Pit05]). This technique has been applied to probabilistic typed higher-order languages by Bizjak and Birkedal [BB15]. Their probabilistic logical relation uses biorthogonality, and is defined on terms, rather than on distributions. These features introduce some universal quantification (e.g., on evaluation contexts) which makes it difficult to prove examples such as 8.43, as discussed in [Biz16, Section 1.5]. Proof techniques combining features of bisimulations and logical relations in the non-probabilistic case are studied in [HDNV12; Nei+15; JT15].

In denotational semantics, fully abstract models for probabilistic PCF have been studied in [GL15] using domain theory and adding statistical termination testers, and in [ETP14] using probabilistic coherence spaces. [DH02] provides a fully abstract game semantics for probabilistic Algol, using a quotienting step.

Finally, in this work we have only considered exact behavioral equivalences, as opposed to approximate behavioral equivalences (allowing the programs to differ up to a certain probability value p) or metrics (measuring the distance between the behaviors of probabilistic programs) [DLT08; DJGP02]. Bisimulation metrics for an affine probabilistic pure λ-calculus have been recently proposed in [CD15]. Applicative bisimulation metric is proved to be sound with respect to the contextual distance, and a metric for an extensions of the language with tuples is defined. In order to be sound with respect to the contextual metric, the tuple distance is endowed with a notion of environment.

9.2 Conclusions and future work

In probabilistic λ-calculi, even in cases where applicative bisimilarity is fully abstract for contextual equivalence, the corresponding simulation may not be fully abstract for the contextual preorder. Pure call-by-value is such an example. We have seen in Chapter 7 that extending the probabilistic call-by-value λ-calculus with a parallel disjunction operator allows us to recover full abstraction with respect to the contextual preorder. The soundness proof is carried out throught Howe's technique enriched with non-trivial 'disentangling' properties for sets of real numbers; the completeness proof is based on the encoding of "logical" tests characterizing probabilistic simulation on RPLTSs.

In Chapter 8 we have studied fully abstract environmental bisimulations for probabilistic pure call-by-name and call-by-value λ-calculi, and for a probabilistic λ-calculus with higher-order, local references. In all the considered calculi, full abstraction for environmental bisimilarity carries over to the corresponding simulation, with a similar proof. This shows a further difference between applicative and environmental (bi)simulations in the probabilistic setting.

While we have tried to respect the general schema of environmental bisimulations, our definitions and results present noticeable technical differences. Some differences, such as the appeal to formal sums, are specific to probabilities. Other differences, however, may be seen as insights into environmental bisimulations that were suggested by the study of probabilities. An example is the distinction between a static and a dynamic environment, which reflects the copying facilities of the language on the terms of the environment. This distinction yields sharper congruence results, which show up well in the imperative λ-calculus: with ordinary environmental bisimulations, bisimilarity is fully substitutive only for values, since for general terms substitutivity holds only for evaluation contexts (see Example 8.42).[12] The example in Section 8.2.4 shows that static environments can also be useful in context closures of 'up-to context' techniques.

[12] As a consequence, in ordinary bisimulation, we can prove that terms M and N are contextually equivalent by showing that $\lambda.M$ and $\lambda.N$ are bisimilar [SKS11].

To understand environmental bisimulations for call-by-value calculi, we have found important the study of the imperative extension. Only in the richer language do various aspects of our definitions find a justification: the use of formal sums (Example 8.43); dynamic environments as formal sums of tuples of values, as opposite to, e.g., tuples of formal sums (Example 8.44); the lifting construct to handle first-order values (Example 8.45). The pure call-by-value calculus has allowed us to present the concepts in a simpler setting, as a stepping stone towards the imperative extension, but seems a rather peculiar language, one in which a number of variations of the definitions collapse.

The dynamic environments are used only for the call-by-value calculi. In general, the form of the bisimulation clauses depends on the features of the calculus. It would be interesting to investigate an abstract formulation of bisimulation, of which the concrete definitions presented in Chapter 8 would be instances. Possible bases for such a framework could be coalgebras [RJ12] or bigraphs [Mil06].

Related to the problems with congruence of applicative bisimulations are also the difficulties with "up-to context" techniques (the usefulness of these techniques in higher-order languages and its problems with applicative bisimulations have been studied by Lassen [Las98]; see also [San97; KW06b; PS12b]). Enhancements of the bisimulation proof method, as up-to techniques, are particularly useful for environmental bisimulations [SKS11] because of the quantification over contexts that appear in the definition and that can make bisimulation proofs tedious. We have explored some basic enhancements, mainly in call-by-name, including new forms of up-to techniques specific to probabilities such as 'up-to lifting'. The study of powerful enhancements goes beyond the scopes of this work. It could be pursued in a number of directions, for instance investigating other forms of up-to and strengthening the 'up-to context' enhancements.

Another interesting direction for future work is the addition of concurrency. A major consequence of this could be the move to semantics that combine probabilities with nondeterminism.

Bibliography

[ABLP16] A. Aristizábal, D. Biernacki, S. Lenglet, and P. Polesiuk. "Environmental Bisimulations for Delimited-Control Operators with Dynamic Prompt Generation". In: *Proc. FSCD'16.* 2016, 9:1–9:17.

[Abr87] S. Abramsky. "Observational Equivalence as a Testing Equivalence". In: *Theoretical Computer Science* 53 (1987), pp. 225–241.

[Abr90] S. Abramsky. "The Lazy Lambda Calculus". In: *Research topics in functional programming.* Ed. by D. A. Turner. Addison-Wesley, 1990, pp. 65–116.

[Acz88] P. Aczel. *Non-Well-Founded Sets.* Vol. 14. CSLI Lecture Notes. Stanford: CSLI, 1988.

[AG98] M. Abadi and A. D. Gordon. "A Bisimulation Method for Cryptographic Protocols". In: *Nordic Journal of Computing* 5 (1998). Preliminary version appeared in Proc. ESOP'98, pp. 267–303.

[AO93] S. Abramsky and C.-H. L. Ong. "Full Abstraction in the Lazy Lambda Calculus". In: *Information and Computation* 105.2 (1993), pp. 159–267.

[Bai98] C. Baier. *On Algorithmic Verification Methods for Probabilistic Systems.* Habilitation Thesis, 1998.

[Bar02] F. Bartels. "GSOS for Probabilistic Transition Systems". In: *Proc. CMCS '02.* Vol. 65. ENTCS 1. Elsevier, 2002, pp. 29–53.

[Bar84] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* Studies in Logic and the Foundations of Mathematics 103. North Holland, 1984.

[BB15] A. Bizjak and L. Birkedal. "Step-Indexed Logical Relations for Probability". In: *Proc. FoSSaCS'15.* 2015, pp. 279–294.

[BDGS16] J. Borgström, U. Dal Lago, A. Gordon, and M. Szymczak. "A lambda-calculus foundation for universal probabilistic programming". In: *Proc. ICFP'16.* 2016, pp. 33–46.

[BDL13] M. Bernardo, R. De Nicola, and M. Loreti. "A Companion of Relating Strong Behavioral Equivalences for Processes with Nondeterminism and Probabilitie". In: *CoRR* abs/1305.0538 (2013).

[BDL14a] M. Bernardo, R. De Nicola, and M. Loreti. "Relating Strong Behavioral Equivalences for Processes with Nondeterminism and Probabilities". In: *Theoretical Computer Science* (2014).

[BDL14b] M. Bernardo, R. De Nicola, and M. Loreti. “Revisiting Trace and Testing Equivalences for Nondeterministic and Probabilistic Processes”. In: *Logical Methods in Computer Science* 10(1:16) (2014), pp. 1–42.

[BDP99] M. Boreale, R. De Nicola, and R. Pugliese. “Basic Observables for Processes”. In: *Information and Computation* 149.1 (1999), pp. 77 –98.

[Ben83] J. Benthem. *Modal Logic and Classical Logic.* Napoli: Bibliopolis, 1983.

[BHR84] S. Brookes, C. Hoare, and A. Roscoe. “A Theory of Communicating Sequential Processes”. In: *Journal of the ACM* 31 (1984), pp. 560–599.

[BIM95] B. Bloom, S. Istrail, and A. Meyer. “Bisimulation Can’t Be Traced”. In: *Journal of the ACM* 42 (1995), pp. 232–268.

[Biz16] A. Bizjak. “On Semantics and Applications of Guarded Recursion”. PhD thesis. Aarhus University, 2016.

[BK00] C. Baier and M. Kwiatkowska. “Domain Equations for Probabilistic Processes”. In: *Mathematical Structures in Computer Science* 10 (2000), pp. 665–717.

[BL13] D. Biernacki and S. Lenglet. “Environmental Bisimulations for Delimited-Control Operators”. In: *Proc. APLAS’13.* Vol. 8301. LNCS. Springer, 2013, pp. 333–348.

[BM16] M. Bernardo and M. Miculan. “Disjunctive Probabilistic Modal Logic is Enough for Bisimilarity on Reactive Probabilistic Systems”. In: *Proc. ICTCS’16.* 2016, pp. 203–220.

[BMOW05] F. Breugel, M. Mislove, J. Ouaknine, and J. Worrell. “Domain Theory, Testing and Simulation for Labelled Markov Processes”. In: *Theoretical Computer Science* 333 (2005), pp. 171–197.

[BS98] M. Boreale and D. Sangiorgi. “Bisimulation in name-passing calculi without matching”. In: *Proc. LICS’98.* IEEE Computer Society Press, 1998.

[BSV14a] M. Bernardo, D. Sangiorgi, and V. Vignudelli. “On the Discriminating Power of Passivation and Higher-Order Interaction”. In: *Proc. CSL-LICS’14.* ACM, 2014.

[BSV14b] M. Bernardo, D. Sangiorgi, and V. Vignudelli. “On the Discriminating Power of Testing Equivalences for Reactive Probabilistic Systems: Results and Open Problems”. In: *Proc. QEST’14.* LNCS 8657. Springer, 2014, pp. 281–296.

[CD14] R. Crubillé and U. Dal Lago. “On Probabilistic Applicative Bisimulation and Call-by-Value λ-Calculi”. In: *Proc. ESOP’14.* Vol. 8410. LNCS. Springer, 2014, pp. 209–228.

[CD15] R. Crubillé and U. Dal Lago. “Metric Reasoning about λ-Terms: The Affine Case”. In: *Proc. LICS’15.* IEEE, 2015, pp. 633–644.

[CDLSV15] R. Crubillé, U. Dal Lago, D. Sangiorgi, and V. Vignudelli. “On Applicative Similarity, Sequentiality, and Full Abstraction”. In: *Correct System Design.* Ed. by R. Meyer, A. Platzer, and H. Wehrheim. LNCS 9360. Springer, 2015, pp. 65–82.

[CVN05] G. Castagna, J. Vitek, and F. Z. Nardelli. “The Seal Calculus”. In: *Information and Computation* 201.1 (2005), pp. 1–54.

[DD07] Y. Deng and W. Du. “Probabilistic Barbed Congruence”. In: *Electronic Notes in Theoretical Computer Science* 190.3 (2007), pp. 185–203.

[DD11] Y. Deng and W. Du. *Logical, Metric, and Algorithmic Characterisations of Probabilistic Bisimulation.* Tech. rep. CMU-CS-11-110. Carnegie Mellon University, 2011.

[Den14] Y. Deng. *Semantics of Probabilistic Processes. An Operational Approach.* Springer-Verlag, 2014.

[DEP02] J. Desharnais, A. Edalat, and P. Panangaden. “Bisimulation for Labelled Markov Processes”. In: *Information and Computation* 179 (2002), pp. 163–193.

[DGHM08] Y. Deng, R. Glabbeek, M. Hennessy, and C. Morgan. “Characterising Testing Preorders for Finite Probabilistic Processes”. In: *Logical Methods in Computer Science* 4(4:4) (2008), pp. 1–33.

[DGHM09] Y. Deng, R. Glabbeek, M. Hennessy, and C. Morgan. “Testing Finitary Probabilistic Processes”. In: *Proc. CONCUR’09.* Vol. 5710. LNCS. Springer, 2009, pp. 274–288.

[DGHMZ07a] Y. Deng, R. Glabbeek, M. Hennessy, C. Morgan, and C. Zhang. “Characterising Testing Preorders for Finite Probabilistic Processes”. In: *Proc. LICS’07.* IEEE-CS Press, 2007, pp. 313–325.

[DGHMZ07b] Y. Deng, R. Glabbeek, M. Hennessy, C. Morgan, and C. Zhang. “Remarks on Testing Probabilistic Processes”. In: *Computation, Meaning, and Logic: Articles Dedicated to Gordon Plotkin.* Vol. 172. ENTCS. Elsevier, 2007, pp. 359–397.

[DH02] V. Danos and R. S. Harmer. “Probabilistic Game Semantics”. In: *ACM Trans. Comput. Logic* 3.3 (2002), pp. 359–382.

[DH84] R. De Nicola and M. Hennessy. “Testing Equivalences for Processes”. In: *Theoretical Computer Science* 34 (1984), pp. 83–133.

[DJGP02] J. Desharnais, R. Jagadeesan, V. Gupta, and P. Panangaden. “The Metric Analogue of Weak Bisimulation for Probabilistic Processes”. In: *Proc. LICS’02.* IEEE Computer Society, 2002, pp. 413–422.

[DL12] P. D’Argenio and M. Lee. “Probabilistic Transition System Specification: Congruence and Full Abstraction of Bisimulation”. In: *Proc. FoSSaCS’12.* Vol. 7213. Lecture Notes in Computer Science. Springer, 2012, pp. 452–466.

[DLSA14] U. Dal Lago, D. Sangiorgi, and M. Alberti. “On Coinductive Equivalences for Higher-order Probabilistic Functional Programs”. In: *Proc. POPL’14.* ACM, 2014, pp. 297–308.

[DLT08] J. Desharnais, F. Laviolette, and M. Tracol. “Approximate Analysis of Probabilistic Processes: Logic, Simulation and Games”. In: *Proc QEST’08.* IEEE Computer Society, 2008, pp. 264–273.

[dP95] U. de'Liguoro and A. Piperno. "Non Deterministic Extensions of Untyped Lambda-Calculus". In: *Information and Computation* 122.2 (1995), pp. 149–177.

[DZ12] U. Dal Lago and M. Zorzi. "Probabilistic Operational Semantics for the Lambda Calculus". In: *RAIRO - Theoretical Informatics and Applications* 46.3 (2012), pp. 413–450.

[ETP14] T. Ehrhard, C. Tasson, and M. Pagani. "Probabilistic Coherence Spaces Are Fully Abstract for Probabilistic PCF". In: *Proc. POPL'14.* San Diego, California, USA, 2014, pp. 309–320.

[GH05] J. Godskesen and T. Hildebrandt. "Extending Howe's Method to Early Bisimulations for Typed Mobile Embedded Resources with Local Names". In: *Proc. FSTTCS'05.* Vol. 3821. Lecture Notes in Computer Science. Springer, 2005, pp. 140–151.

[GL15] J. Goubault-Larrecq. "Full abstraction for non-deterministic and probabilistic extensions of PCF I: The angelic cases". In: *Journal of Logical and Algebraic Methods in Programming* 84.1 (2015), pp. 155 –184.

[Gla01] R. Glabbeek. "The Linear Time - Branching Time Spectrum I. The Semantics of Concrete, Sequential Processes". In: *Handbook of Process Algebra.* Ed. by J. A. Bergstra, A. Ponse, and S. Smolka. Amsterdam: Elsevier, 2001, pp. 3–99.

[Goo13] N. D. Goodman. "The principles and practice of probabilistic programming". In: *Proc. POPL'13.* 2013, pp. 399–402.

[GV92] J. Groote and F. Vaandrager. "Structured Operational Semantics and Bisimulation as a Congruence". In: *Information and Computation* 100 (1992), pp. 202–260.

[HDNV12] C. Hur, D. Dreyer, G. Neis, and V. Vafeiadis. "The marriage of bisimulations and Kripke logical relations". In: *Proc. POPL'12.* 2012, pp. 59–72.

[Hen12] M. Hennessy. "Exploring probabilistic bisimulations, Part I". In: *Formal Aspects of Computing* 24 (2012), pp. 749–768.

[HM85] M. Hennessy and R. Milner. "Algebraic Laws for Nondeterminism and Concurrency". In: *Journal of the ACM* 32.1 (1985), pp. 137–161.

[How89] D. Howe. "Equality In Lazy Computation Systems". In: *Proc. LICS'89.* IEEE Computer Society, 1989, pp. 198–203.

[How96] D. Howe. "Proving Congruence of Bisimulation in Functional Programming Languages". In: *Information and Computation* 124.2 (1996), pp. 103–112.

[HT92] D. Huynh and L. Tian. "On Some Equivalence Relations for Probabilistic Processes". In: *Fundamenta Informaticae* 17 (1992), pp. 211–234.

[JHSY94] B. Jonsson, C. Ho-Stuart, and W. Yi. "Testing and Refinement for Nondeterministic and Probabilistic Processes". In: *Proc. FTRTFT'94.* Vol. 863. LNCS. Springer, 1994, pp. 418–430.

[JL91] B. Jonsson and K. G. Larsen. “Specification and Refinement of Probabilistic Processes”. In: *Proc. LICS’91.* IEEE Computer Society, 1991, pp. 266–277.

[Jon90] C. Jones. “Probabilistic Non-Determinism”. PhD thesis. University of Edinburgh, 1990.

[JPR09] R. Jagadeesan, C. Pitcher, and J. Riely. “Open Bisimulation for Aspects”. In: *T. Aspect-Oriented Software Development* 5 (2009), pp. 72–132.

[JR12] B. Jacobs and J. Rutten. “An Introduction to (co)algebra and (co)induction”. In: *Advanced Topics in Bisimulation and Coinduction.* Ed. by D. Sangiorgi and J. Rutten. Vol. 52. Cambridge Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press, 2012. Chap. 2, pp. 38–99.

[JR99] A. Jeffrey and J. Rathke. “Towards a Theory of Bisimulation for Local Names”. In: *Proc. LICS’99.* 1999, pp. 56–66.

[JS90] C.-C. Jou and S. Smolka. “Equivalences, Congruences, and Complete Axiomatizations for Probabilistic Processes”. In: *Proc. CONCUR’90.* Vol. 458. LNCS. Springer, 1990, pp. 367–383.

[JT15] G. Jaber and N. Tabareau. “Kripke Open Bisimulation - A Marriage of Game Semantics and Operational Techniques”. In: *Proc. APLAS’15.* 2015, pp. 271–291.

[JY95] B. Jonsson and W. Yi. “Compositional Testing Preorders for Probabilistic Processes”. In: *Proc. LICS’95.* IEEE-CS Press, 1995, pp. 431–441.

[KH13] V. Koutavas and M. Hennessy. “Symbolic Bisimulation for a Higher-Order Distributed Language with Passivation”. In: *Proc. CONCUR’13.* Vol. 8052. Lecture Notes in Computer Science. Springer, 2013, pp. 167–181.

[KLS11] V. Koutavas, P. Levy, and E. Sumii. “From Applicative to Environmental Bisimulation”. In: *Electronic Notes in Theoretical Computer Science* 276 (2011), pp. 215–235.

[KN98] M. Kwiatkowska and G. Norman. “A Testing Equivalence for Reactive Probabilistic Processes”. In: *Proc. EXPRESS’98.* Vol. 16(2). ENTCS. Elsevier, 1998, pp. 114–132.

[KW06a] V. Koutavas and M. Wand. “Bisimulations for Untyped Imperative Objects”. In: *Proc. ESOP’06.* 2006, pp. 146–161.

[KW06b] V. Koutavas and M. Wand. “Small Bisimulations for Reasoning About Higher-Order Imperative Programs”. In: *Proc. POPL’06.* 2006, pp. 141–152.

[Las98] S. B. Lassen. “Relational Reasoning about Functions and Nondeterminism”. PhD thesis. University of Aarhus, 1998.

[LL07] S. B. Lassen and P. B. Levy. “Typed Normal Form Bisimulation”. In: *Proc. CSL’07.* Vol. 4646. LNCS. Springer, 2007, pp. 283–297.

[LS91] K. Larsen and A. Skou. “Bisimulation Through Probabilistic Testing”. In: *Information and Computation* 94 (1991), pp. 1–28.

[LSS09a] S. Lenglet, A. Schmitt, and J.-B. Stefani. "Howe's Method for Calculi with Passivation". In: *Proc. CONCUR'09*. Vol. 5710. Lecture Notes in Computer Science. Springer, 2009, pp. 448–462.

[LSS09b] S. Lenglet, A. Schmitt, and J.-B. Stefani. "Normal Bisimulations in Calculi with Passivation". In: *Proc. FoSSaCS'09*. Vol. 5504. Lecture Notes in Computer Science. Springer, 2009, pp. 257–271.

[LSS11] S. Lenglet, A. Schmitt, and J.-B. Stefani. "Characterizing Contextual Equivalence in Calculi with Passivation". In: *Information and Computation* 209.11 (2011), pp. 1390–1433.

[LT09] R. Lanotte and S. Tini. "Probabilistic bisimulation as a congruence". In: *ACM Transactions on Computational Logic* 10.2 (2009).

[Mil06] R. Milner. "Pure bigraphs: Structure and dynamics." In: *Inf. Comput.* 204.1 (2006), pp. 60–122.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Ed. by G. Goos and J. Hartmanis. Vol. 92. Lecture Notes in Computer Science. Berlin: Springer, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mit96] J. C. Mitchell. *Foundations for Programming Languages.* MIT Press, 1996.

[Mor68] J. Morris. "Lambda-calculus models of programming languages". PhD thesis. Massachusetts Institute of Technology., 1968.

[MOTW99] J. Maraist, M. Odersky, D. N. Turner, and P. Wadler. "Call-by-name, Call-by-value, Call-by-need and the Linear lambda Calculus". In: *Theoretical Computer Science* 228.1-2 (1999), pp. 175–210.

[MS92] R. Milner and D. Sangiorgi. "Barbed Bisimulation". In: *Proc. ICALP'92.* Springer, 1992, pp. 685–695.

[MTHM97] R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The Definition of Standard ML (Revised).* MIT Press, 1997.

[Nei+15] G. Neis et al. "Pilsner: A Compositionally Verified Compiler for a Higher-order Imperative Language". In: *Proc. ICFP'15.* New York, NY, USA: ACM, 2015, pp. 166–178.

[Nic87] R. D. Nicola. "Extensional Equivalences for Transition Systems". In: *Acta Informatica* 24 (1987), pp. 211–237.

[Ong93] C.-H. L. Ong. "Non-Determinism in a Functional Setting". In: *Proc. LICS'93.* 1993, pp. 275–286.

[Par81] D. Park. "Concurrency and Automata on Infinite Sequences". In: *Proc. 5th GI Conference on Theoretical Computer Science.* Vol. 104. LNCS. Springer, 1981, pp. 167–183.

[Pfe01] A. Pfeffer. "IBAL: A Probabilistic Rational Programming Language". In: *IJCAI.* Morgan Kaufmann, 2001, pp. 733–740.

[Phi87] I. Phillips. "Refusal Testing". In: *Theoretical Computer Science* 50 (1987), pp. 241–284.

[Pie02] B. C. Pierce. *Types and Programming Languages.* MIT Press, 2002.

[Pit05] A. Pitts. “Typed Operational Reasoning”. In: *Advanced Topics in Types and Programming Languages.* Ed. by B. C. Pierce. MIT Press, 2005. Chap. 7, pp. 245–289.

[Pit12] A. Pitts. “Howe’s Method for Higher-order Languages”. In: *Advanced Topics in Bisimulation and Coinduction.* Ed. by D. Sangiorgi and J. Rutten. Vol. 52. Cambridge Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press, 2012. Chap. 5, pp. 197–232.

[Plo77] G. D. Plotkin. “LCF Considered as a Programming Language”. In: *Theor. Comput. Sci.* 5.3 (1977), pp. 223–255.

[PPT08] S. Park, F. Pfenning, and S. Thrun. “A probabilistic language based on sampling functions”. In: *ACM Transactions on Programming Languages and Systems* 31.1 (2008).

[PS11] A. Piérard and E. Sumii. “Sound Bisimulations for Higher-Order Distributed Process Calculus”. In: *Proc. FoSSaCS’11.* 2011, pp. 123–137.

[PS12a] A. Piérard and E. Sumii. “A Higher-Order Distributed Calculus with Name Creation”. In: *Proc. LICS’12.* 2012, pp. 531–540.

[PS12b] D. Pous and D. Sangiorgi. “Enhancements of the bisimulation proof method”. In: *Advanced Topics in Bisimulation and Coinduction.* Ed. by D. Sangiorgi and J. Rutten. Cambridge University Press, 2012.

[RJ12] J. Rutten and B. Jacobs. “(Co)Algebras and (Co)Induction”. In: *Advanced Topics in Bisimulation and Coinduction.* Ed. by D. Sangiorgi and J. Rutten. Cambridge University Press, 2012.

[RP02] N. Ramsey and A. Pfeffer. “Stochastic lambda calculus and monads of probability distributions”. In: *Proc. POPL’02.* 2002, pp. 154–165.

[San12a] D. Sangiorgi. *Introduction to Bisimulation and Coinduction.* Cambridge: Cambridge University Press, 2012.

[San12b] D. Sangiorgi. “On the origins of bisimulation and coinduction”. In: *Advanced Topics in Bisimulation and Coinduction.* Ed. by D. Sangiorgi and J. Rutten. Vol. 52. Cambridge Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press, 2012. Chap. 1, pp. 1–37.

[San92] D. Sangiorgi. “Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms”. PhD thesis CST–99–93. Department of Computer Science, University of Edinburgh, 1992.

[San94] D. Sangiorgi. “The Lazy Lambda Calculus in a Concurrency Scenario”. In: *Information and Computation* 111.1 (May 1994), pp. 120–153.

[San97] D. Sands. “From SOS Rules to Proof Principles: An Operational Metatheory for Functional Languages”. In: *Proc. POPL’97.* 1997, pp. 428–441.

[SD78] N. Saheb-Djahromi. “Probabilistic LCF”. In: *Proc. MFCS’78.* Vol. 64. LNCS. 1978, pp. 442–451.

[Seg95] R. Segala. “Modeling and Verification of Randomized Distributed Real-Time Systems”. PhD thesis. MIT, 1995.

[Seg96] R. Segala. “Testing Probabilistic Automata”. In: *Proc. CONCUR’96.* LNCS 1119. Springer, 1996, pp. 299–314.

[Sew+07] P. Sewell et al. “Acute: High-level Programming Language Design for Distributed Computation”. In: *Journal of Functional Programming* 17.4-5 (2007), pp. 547–612.

[SKS07] D. Sangiorgi, N. Kobayashi, and E. Sumii. “Logical Bisimulations and Functional Languages”. English. In: *Proc. FSEN’07.* LNCS 4767. Springer, 2007, pp. 364–379.

[SKS11] D. Sangiorgi, N. Kobayashi, and E. Sumii. “Environmental Bisimulations for Higher-Order Languages”. In: *ACM Transactions on Programming Languages and Systems* 33.1:5 (2011).

[SL07] K. Støvring and S. B. Lassen. “A Complete, Co-inductive Syntactic Theory of Sequential Control and State”. In: *Proc. POPL’07.* 2007, pp. 161–172.

[SL95] R. Segala and N. Lynch. “Probabilistic simulations for probabilistic processes”. In: *Nordic Journal of Computing* 2.2 (1995), pp. 250–273.

[SP07a] E. Sumii and B. C. Pierce. “A Bisimulation for Dynamic Sealing”. In: *Theoretical Computer Science* 375.1-3 (2007). A preliminary version in *Proc. POPL’04*, 2004, pp. 169–192.

[SP07b] E. Sumii and B. C. Pierce. “A bisimulation for type abstraction and recursion”. In: *Journal of the ACM* 54.5 (2007). A preliminary version in *Proc. POPL’05*, 2005.

[SS03] A. Schmitt and J.-B. Stefani. “The M-calculus: a Higher-Order Distributed Process Calculus”. In: *Proc. POPL’03.* ACM, 2003, pp. 50–61.

[SS05] A. Schmitt and J.-B. Stefani. “The Kell Calculus: A Family of Higher-Order Distributed Process Calculi”. In: *Proc. GC’04.* Vol. 3267. LNCS. Springer, 2005, pp. 146–178.

[SS09] N. Sato and E. Sumii. “The Higher-Order, Call-by-Value Applied Pi-Calculus”. In: *Proc. APLAS’09.* 2009, pp. 311–326.

[SV04] A. Sokolova and E. Vink. “Probabilistic Automata: System Types, Parallel Composition and Comparison”. In: *Validation of Stochastic Systems.* Vol. 2925. LNCS. Springer, 2004, pp. 1–43.

[SV16] D. Sangiorgi and V. Vignudelli. “Environmental Bisimulations for Probabilistic Higher-order Languages”. In: *Proc. POPL’16.* ACM, 2016, pp. 595–607.

[SYWHK16] S. Staton, H. Yang, F. Wood, C. Heunen, and O. Kammar. “Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints”. In: *Proc. LICS’16.* 2016, pp. 525–534.

[Tho93] B. Thomsen. “Plain CHOCS, A Second Generation Calculus for Higher-Order Processes”. In: *Acta Informatica* 30 (1993), pp. 1–59.

[YL92] W. Yi and K. Larsen. “Testing Probabilistic and Nondeterministic Processes”. In: *Proc. PSTV’92.* North-Holland, 1992, pp. 47–61.