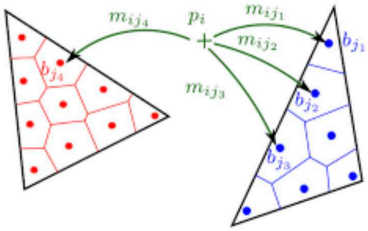
AGENCE NATIONALE DE LA RECHERCHE

ANR

REPAS

RELIABLE AND PRIVACY-AWARE SOFTWARE SYSTEMS



Deliverable D4.b

MODULAR COINDUCTION UP-TO FOR HIGHER-ORDER LANGUAGES VIA FIRST-ORDER TRANSITION SYSTEMS

JEAN-MARIE MADIOT, DAMIEN POUS, AND DAVIDE SANGIORGI

INRIA, France

Plume team, LIP, CNRS, ENS Lyon, Université de Lyon, France

Università di Bologna, Italy; INRIA, France

Abstract. The bisimulation proof method can be enhanced by employing *'bisimulations up-to'* techniques. A comprehensive theory of such enhancements has been developed for first-order (i.e., CCS-like) labelled transition systems (LTSs) and bisimilarity, based on abstract fixed-point theory and compatible functions.

We transport this theory onto languages whose bisimilarity and LTS go beyond those of first-order models. The approach consists in exhibiting fully abstract translations of the more sophisticated LTSs and bisimilarities onto the first-order ones. This allows us to reuse directly the large corpus of up-to techniques that are available on first-order LTSs. The only ingredient that has to be manually supplied is the compatibility of basic up-to techniques that are specific to the new languages. We investigate the method on the π-calculus, the λ-calculus, and a (call-by-value) λ-calculus with references.

1. Introduction

One of the keys for the success of bisimulation is its associated proof method, whereby to prove two terms equivalent, one exhibits a relation containing the pair and one proves it to be a bisimulation. The bisimulation proof method can be enhanced by employing relations called *'bisimulations up-to'* [33, 20, 26, 30]; see [29] for a historical perspective. These need not be bisimulations; they are simply *contained in* a bisimulation. Such techniques have been widely used in languages for mobility such as π-calculus or higher-order languages such as the λ-calculus, or Ambients (e.g., [16, 22, 36]).

Several forms of bisimulation enhancements have been introduced: 'bisimulation up to bisimilarity' [23] where the derivatives obtained when playing bisimulation games can be rewritten using bisimilarity itself; 'bisimulation up to transitivity' where the derivatives may be rewritten using the up-to relation [33]; 'bisimulation up to context' [32], where a common context may be removed from matching derivatives. Further enhancements may exploit the peculiarities of the definition of bisimilarity on certain classes of languages: e.g., the up-to-injective-substitution techniques of the π-calculus [12, 36], techniques for shrinking or enlarging the environment in languages with information hiding mechanisms (e.g., existential types, encryption and decryption constructs [1, 39, 40]), frame equivalence in the psi-calculi [24], or higher-order languages [15, 17]. Lastly, it is important to notice

Preprint submitted to
Logical Methods in Computer Science

that one often wishes to use *combinations* of up-to techniques. For instance, up-to-context alone does not appear to be very useful; its strength comes out in association with other techniques, such as up-to-bisimilarity or up-to-transitivity.

The main problem with up-to techniques is proving their soundness (i.e. ensuring that any 'bisimulation up-to' is contained in bisimilarity). In particular, the proofs of complex combinations of techniques can be difficult or, at best, long and tedious. Moreover, if one modifies the language or the up-to technique, the entire proof has to be redone from scratch. Indeed the soundness of some up-to techniques is quite fragile, and may break when such variations are made. For instance, up-to-bisimilarity usually fails for weak bisimilarity, and in certain languages the combination of up-to-bisimilarity and up-to-context fails while the two techniques are sound when taken separately.

This problem has been the motivation for the development of a theory of enhancements, summarised in [26]. Expressed in the general fixed-point theory on complete lattices, this theory has been fully developed for both strong and weak bisimilarity, in the case of first-order labelled transition systems (LTSs) where transitions represent pure synchronisations among processes. In this framework, up-to techniques are represented using *compatible* functions, whose class enjoys nice algebraic properties (an earlier variant, with similar properties, is that of *respectful* functions [33]). This allows one to derive complex up-to techniques algebraically, by composing simpler techniques by means of a few operators.

Only a small part of the theory has been transported onto other forms of transition systems, on a case by case basis. Transferring the whole theory would be a substantial and non-trivial effort. Moreover it might have limited applicability, as this work would probably have to be based on specific shapes for transitions and bisimilarity (a wide range of variations exist, e.g., in higher-order languages).

Here we explore a different approach to the transport of the theory of bisimulation enhancements onto richer languages. The approach consists in exhibiting fully abstract translations of the more sophisticated LTSs and bisimilarities onto first-order LTSs and bisimilarity. This allows us to import directly the existing theory for first-order bisimulation enhancements onto the new languages. Most importantly, the schema allows us to combine up-to techniques for the richer languages. The only additional ingredient that has to be provided manually is the soundness of some up-to techniques that are specific to the new languages. This typically includes the up-to-context techniques, since those contexts are not first-order.

Our hope is that the method proposed here will make it possible to obtain a single formalised library about up-to techniques, that can be reused for a wide range of calculi: currently, all existing formalisations of such techniques in a proof assistant are specific to a given calculus: π-calculus [8, 10], the psi-calculi [24], or a miniML language [11].

We consider three languages in this paper: the π-calculus, the call-by-name λ-calculus, and an imperative call-by-value λ-calculus (a call-by-value λ-calculus with references). We focus on weak bisimilarity, whose theory is more involved than that of strong bisimilarity: some the congruence properties break or require subtle proofs, and these differences between the strong and weak case are magnified when it comes to enhancements of the bisimulation proof method.

When we translate a transition system into a first-order one, the grammar for the transition labels can be complex (e.g. include terms, labels, or contexts). What nevertheless makes these systems 'first-order' is that these labels are taken as syntactic atomic objects, that may only be checked for syntactic equality. In other words, the bisimulation games

which we play on those first-order LTS are just the plain and standard ones, without any side conditions on free names, alpha-conversion, or semantical comparison of higher-order values. Leifer and Milner [19], using contexts as first-order labels, derive, from any appropriate reactive system, a first-order LTS for which strong bisimilarity is a congruence; this approach however does not handle weak bisimilarity or up-to techniques.

Full abstraction of a translation does not imply that all desirable or expected up-to techniques come for free: sometimes the translations have to be designed with care to be useful. We shall see this with the π-calculus, where *early bisimilarity* can be handled properly, but where the natural and fully abstract adaptation of the translation for *late bisimilarity* does not provide us with satisfactory up-to techniques (see Remark 16). In the same manner, our translation for the λ-calculus has similarities with the first-order translation of environmental bisimulation in [14], but the latter, while fully abstract, would disallow important up-to techniques (see Remark 32).

Forms of up-to-context have already been derived for the languages we consider in this paper [16, 34, 36]. The corresponding soundness proofs are difficult (especially in λ-calculi), and require a mix of induction (on contexts) and coinduction (to define bisimulations). Recasting up-to-context within the theory of bisimulation enhancements has several advantages. First, this allows us to combine this technique with other techniques, directly. Second, congruence (or substitutivity) of bisimilarity becomes a corollary of the compatibility of the up-to-context function (in higher-order languages these two kinds of proofs are usually hard and very similar). And third, this allows us to decompose the up-to-context function into smaller pieces, essentially one for each operator of the language, yielding more modular proofs, also allowing, if needed, to rule out those contexts that do not preserve bisimilarity (e.g., input prefix in the π-calculus).

The translation of the π-calculus LTS into a first-order LTS follows the schema of abstract machines for the π-calculus (e.g., [41]) in which the issue of the choice of fresh names is resolved by ordering the names and indexing the processes with a name that represents an upper bound to the names occurring in the process. Various forms of bisimulation enhancements have appeared in papers on the π-calculus or dialects of it. A translation of higher-order π-calculi into first-order processes has been proposed by Koutavas et al. [13]. While the shape of our translations of λ-calculi is similar, our LTSs differ since they are designed to recover the theory of bisimulation enhancements. In particular, using the fully abstract LTSs from [13], does not make it possible to use up-to context techniques (essentially by the same argument as the one in Remark 32). In the λ-calculus, limited forms of up-to techniques have been developed for applicative bisimilarity, where the soundness of up-to-context is still an open problem [16, 17]. More powerful versions of up-to-context exist for forms of bisimilarity on open terms; e.g., open bisimilarity or head-normal-form bisimilarity [18]. Currently, the form of bisimilarity for closed higher-order terms that allows the richest range of up-to techniques is environmental bisimilarity [14, 34]. However, even in this setting, the proofs of combinations of up-to techniques are usually long and non-trivial. Our translation of higher-order terms to first-order terms is designed to recover environmental bisimilarity and to simplify the tasks of proving up-to techniques and of combining them.

Open bisimilarity or normal-form bisimilarities have also been used to avoid quantification over function arguments [31, 16, 15], and powerful and compositional approaches to up-to techniques have been developed in such settings [5, 6]. Yet the resulting LTSs still manipulate binders and require fresh instantiations of them, much in the same way as the

π-calculus does; as such, as discusssed above, the theory of first-order LTSs is not directly applicable.

This paper is an extended version of the conference paper [21], with all proofs. That paper introduced a notion 'compatibility up-to' in order to assemble several up-to techniques whose soundness depends on each other, in a 'mutually coinductive' way. This led to the development of the notion of "companion" [28], reviewed in Section 2, as a cleaner way of doing the same thing and more. We rewrote the current paper to make use of the companion in place of 'compatibility up-to'. This also allowed us to established novel results for additional proof refactoring (Lemmas 22 and 28).

In Section 2 we review the theory of first-order bisimulation and up-to techniques. In Sections 3 to 5 we treat the π-calculus, the (pure) call-by-name λ-calculus, and the imperative call-by-value λ-calculus, respectively. In Section 6, we show an example of how the wide spectrum of up-to techniques made available via our translations allows us to simplify relations needed in bisimilarity proofs, facilitating their description and reducing their size.

Notations. We let $\mathcal{R}, \mathcal{S}$ range over binary relations and we often write $x\ \mathcal{R}\ y$ for $(x, y) \in \mathcal{R}$. Given two relations $\mathcal{R}, \mathcal{S}$, we write $\mathcal{RS}$ for their relational composition, i.e., $\mathcal{RS} \triangleq \{(x, z) \mid \exists y, x\ \mathcal{R}\ y \wedge y\ \mathcal{S}\ z\}$, $\mathcal{R}^+$ for the transitive closure of $\mathcal{R}$, and $\mathcal{R}^*$ for its reflexive transitive closure. We use the standard arrow notation $\mapsto$ to denote functions when the domain is clear, and $\Rightarrow$ for logical implication; other arrow notations will be introduced as we go along.

In languages defined from a grammar, a *context* C of arity $n \in \mathbb{N}$ is a term with numbered holes $[\cdot]_1, \ldots, [\cdot]_n$, where each hole $[\cdot]_i$ can appear any number of times in C. We write $C[P_1, \ldots, P_n]$ for the application of such a context to n terms $P_1, \ldots, P_n$ of the language.

2. First-order bisimulation and up-to techniques

As explained in the introduction, the results in this section are not new: we review general-purpose tools that we exploit to prove soundness of up-to techniques. These tools were obtained in several steps: respectfulness is from [33]; we refined it into the notion of *compatibility up-to* in the conference version of this paper [21], and this refinement eventually led to the notion of the companion [28] and to the associated tools we exploit here.

A *first-order Labelled Transition System*, briefly LTS, is a triple $(Pr, Act, \longrightarrow)$ where Pr is a non-empty set of states (or processes), Act is the set of *actions* (or *labels*), and $\longrightarrow \subseteq Pr \times Act \times Pr$ is the *transition relation*. We use P, Q, R to range over the processes of the LTS, and μ to range over the labels in Act, and, as usual, write $P \xrightarrow{\mu} Q$ when $(P, \mu, Q) \in \longrightarrow$. We assume that Act includes a special action τ that represents an internal activity of the processes. We derive bisimulation from the notion of *progression* between relations.

Definition 1. We define the monotone function **sp** on relations on processes of an LTS:

$$\begin{aligned}\mathbf{sp}(\mathcal{R}) \triangleq \{(P, Q) \mid\ & \left(\forall P'\ \forall \mu \quad P \xrightarrow{\mu} P' \ \Rightarrow\ \exists Q'\ Q \xrightarrow{\mu} Q' \wedge P'\ \mathcal{R}\ Q'\right) \\ & \wedge \left(\forall Q'\ \forall \mu \quad Q \xrightarrow{\mu} Q' \ \Rightarrow\ \exists P'\ P \xrightarrow{\mu} P' \wedge P'\ \mathcal{R}\ Q'\right)\}\end{aligned}$$

We say that $\mathcal{R}$ *strongly progresses to* $\mathcal{S}$, written $\mathcal{R} \underset{\mathbf{sp}}{\rightsquigarrow} \mathcal{S}$, if $\mathcal{R} \subseteq \mathbf{sp}(\mathcal{S})$. A relation $\mathcal{R}$ is a *strong bisimulation* if $\mathcal{R} \underset{\mathbf{sp}}{\rightsquigarrow} \mathcal{R}$; and *strong bisimilarity*, $\sim$, is the union of all strong bisimulations.

To define weak progression we need weak transitions, defined as usual: first, $P \xrightarrow{\hat{\mu}} P'$ means $P \xrightarrow{\mu} P'$ or ($\mu = \tau$ and $P = P'$); and $\stackrel{\hat{\mu}}{\Longrightarrow}$ is $\Longrightarrow\xrightarrow{\hat{\mu}}\Longrightarrow$ where $\Longrightarrow$ is the reflexive transitive closure of $\xrightarrow{\tau}$.

Definition 2. We define the monotone function $\mathbf{wp}$ on relations on processes of an LTS:

$$\mathbf{wp}(\mathcal{R}) \triangleq \{(P,Q) \mid \Big(\forall P'\ \forall \mu \quad P \xrightarrow{\mu} P' \ \Rightarrow\ \exists Q'\ Q \stackrel{\hat{\mu}}{\Longrightarrow} Q' \wedge P'\ \mathcal{R}\ Q'\Big) \\ \wedge \Big(\forall Q'\ \forall \mu \quad Q \xrightarrow{\mu} Q' \ \Rightarrow\ \exists P'\ P \stackrel{\hat{\mu}}{\Longrightarrow} P' \wedge P'\ \mathcal{R}\ Q'\Big)\}$$

We say that $\mathcal{R}$ *weakly progresses to* $\mathcal{S}$, written $\mathcal{R} \underset{\mathbf{wp}}{\rightsquigarrow} \mathcal{S}$, if $\mathcal{R} \subseteq \mathbf{wp}(\mathcal{S})$. A relation $\mathcal{R}$ is a *weak bisimulation* if $\mathcal{R} \underset{\mathbf{wp}}{\rightsquigarrow} \mathcal{R}$; and *weak bisimilarity*, $\approx$, is the union of all weak bisimulations.

Below we summarise the ingredients of the theory of bisimulation enhancements for first-order LTSs from [26] that will be needed in the sequel. We use f and g to range over monotone functions on relations over a fixed set of states. Each such function represents a potential up-to technique; only the *sound* functions, however, qualify as up-to techniques:

Definition 3. A function f is *sound for* $\sim$ if $\mathcal{R} \underset{\mathbf{sp}}{\rightsquigarrow} f(\mathcal{R})$ implies $\mathcal{R} \subseteq \sim$, for all $\mathcal{R}$; similarly, f is *sound for* $\approx$ if $\mathcal{R} \underset{\mathbf{wp}}{\rightsquigarrow} f(\mathcal{R})$ implies $\mathcal{R} \subseteq \approx$, for all $\mathcal{R}$.

Unfortunately, the class of sound functions does not enjoy good algebraic properties. In particular, the composition and the pairwise union of two sound functions are not necessarily sound [26, Section 6.3.3]. As a remedy to this, the subset of *compatible* functions has been proposed. The concepts in the remainder of the section can be instantiated with both strong and weak bisimilarities; we thus use $\mathbf{p}$ to range over $\mathbf{sp}$ or $\mathbf{wp}$.

Definition 4. We write $f \overset{\mathbf{p}}{\rightsquigarrow} g$ when $f \circ \mathbf{p} \subseteq \mathbf{p} \circ g$. A monotone function f on relations is $\mathbf{p}$-*compatible* if $f \overset{\mathbf{p}}{\rightsquigarrow} f$.

In other terms, $f \overset{\mathbf{p}}{\rightsquigarrow} g$ when $\mathcal{R} \underset{\mathbf{p}}{\rightsquigarrow} \mathcal{S}$ implies $f(\mathcal{R}) \underset{\mathbf{p}}{\rightsquigarrow} g(\mathcal{S})$ for all $\mathcal{R}$ and $\mathcal{S}$.

Lemma 5. If f is $\mathbf{sp}$-compatible, then f is sound for $\sim$; if f is $\mathbf{wp}$-compatible, then f is sound for $\approx$.

Simple examples of compatible functions are the identity function id and the function mapping any relation onto bisimilarity (strong or weak case, depending on the considered case). This means that $(\mathcal{R} \mapsto \sim)$ is $\mathbf{sp}$-compatible, and $(\mathcal{R} \mapsto \approx)$ is $\mathbf{wp}$-compatible. In addition, $(\mathcal{R} \mapsto \sim)$ is also a useful $\mathbf{wp}$-compatible function. The class of compatible functions is closed under function composition and union (where the union $\cup F$ of a set of functions F is the point-wise union mapping $\mathcal{R}$ to $\bigcup_{f \in F} f(\mathcal{R})$), and thus under ω-iteration (where the ω-iteration f^ω of a function f maps $\mathcal{R}$ to $\bigcup_{n \in \mathbb{N}} f^n(\mathcal{R})$). For example $(\mathcal{R} \mapsto (\mathcal{R} \cup \sim))$ is $\mathbf{sp}$- and $\mathbf{wp}$-compatible.

Other examples of compatible functions are typically contextual closure functions, or *up-to-context*, mapping a relation into its closure w.r.t. a given set of contexts. Not all

context closures are compatible: their compatibility must be established separately for each LTS. For such functions, the following lemma shows that the compatibility of up-to-context implies the congruence of (strong or weak) bisimilarity.

Lemma 6. If f is **sp**-compatible, then $f(\sim) \subseteq \sim$; similarly if f is **wp**-compatible, then $f(\approx) \subseteq \approx$.

Certain closure properties for compatible functions however only hold in the strong case. The main example is the *chaining operator* $\frown$, which implements pointwise relational composition:

$$f \frown g\ (\mathcal{R}) \triangleq f(\mathcal{R})\ g(\mathcal{R})$$

where the juxtaposition $\mathcal{R}\ \mathcal{S}$ of two relations $\mathcal{R}$ and $\mathcal{S}$ and denotes their relational composition. Using chaining we can obtain the compatibility of the 'up-to-transitivity' function, mapping a relation $\mathcal{R}$ onto its reflexive and transitive closure $\mathcal{R}^\star$. Another important example of compatible function in the strong case is 'up-to-strong-bisimilarity' ($\mathcal{R} \mapsto \sim\mathcal{R}\sim$), which is also compatible in the weak case.

In contrast, the counterpart of this latter function in the weak case, $\mathcal{R} \mapsto \approx\mathcal{R}\approx$, is unsound. This is a major drawback in up-to techniques for weak bisimilarity, which can be partially overcome by resorting to the *expansion* relation $\gtrsim$ [4, 35] (a refinement of expansion is the contraction relation [37]). Expansion is an asymmetric refinement of weak bisimilarity whereby $P \gtrsim Q$ holds if P and Q are bisimilar and, in addition, Q is at least as efficient as P, in the sense that Q is capable of producing the same activity as P without ever performing more internal activities (the τ-actions). More precisely, the associated progression function is **ep** defined below, and $\gtrsim$ is the union of all $\mathcal{R}$ such that $\mathcal{R} \subseteq \mathbf{ep}(\mathcal{R})$.

$$\begin{aligned}\mathbf{ep}(\mathcal{R}) \triangleq \{(P,Q) \mid\ & \left(\forall P'\ \forall \mu \quad P \xrightarrow{\mu} P' \ \Rightarrow\ \exists Q'\ Q \xrightarrow{\hat{\mu}} Q' \wedge P'\ \mathcal{R}\ Q'\right) \\ & \wedge \left(\forall Q'\ \forall \mu \quad Q \xrightarrow{\mu} Q' \ \Rightarrow\ \exists P'\ P \Longrightarrow\!\!\xrightarrow{\mu}\!\!\Longrightarrow P' \wedge P'\ \mathcal{R}\ Q'\right)\}\end{aligned}$$

Up-to-expansion yields a function ($\mathcal{R} \mapsto \gtrsim\mathcal{R}\lesssim$) that is contained in a **wp**-compatible function, and which can be freely combined with any **wp**-compatible function, yielding for instance the 'up-to-expansion-and-contexts' technique. More sophisticated up-to techniques can be obtained by carefully adjusting the interplay between visible and internal transitions, and by taking into account termination hypotheses [26].

Some further compatible functions are the functions **sp** and **wp** themselves (indeed a function f is **p**-compatible if $f \circ \mathbf{p} \subseteq \mathbf{p} \circ f$, hence trivially f can be replaced with **p** itself). Intuitively, the use of **sp** and **wp** as up-to techniques means that, in a diagram-chasing argument, the two derivatives need not be related; it is sufficient that the derivatives of such derivatives be related. Accordingly, we sometimes call functions **sp** and **wp** *unfolding* functions. We will use **sp** in the example in Section 6 and **wp** in Sections 4 and 5, when proving the **wp**-compatibility of the up-to-context techniques. Note that up-to-context functions are the only ones that need to be proved compatible separately for each LTS; in this section all other functions mentioned as compatible are so for every first-order LTS.

2.1. **Companion.** We say that f is *below* g, and write $f \subseteq g$, when $f(\mathcal{R}) \subseteq g(\mathcal{R})$ for every relation $\mathcal{R}$. Any function below a sound function is sound as well. Similarly, if $f \subseteq g$ and $g(\sim) \subseteq \sim$ then $f(\sim) \subseteq \sim$.

In general a function below a compatible function need not be itself compatible. However, it turns out that there is a largest compatible function, which is called the *companion* of $\mathbf{p}$ [28], defined as the pointwise union of all $\mathbf{p}$-compatible functions:

$$t_{\mathbf{p}} \triangleq \bigcup_{f \overset{\mathbf{p}}{\rightsquigarrow} f} f$$

In the following, we generally omit the subscript or superscript $\mathbf{p}$ when clear from the context. Since t is itself compatible we can deduce from Lemmas 5 and 6 that if $f \subseteq t_{\mathbf{sp}}$, then f is sound for $\sim$ and $f(\sim) \subseteq \sim$. Similarly in the weak case: if $f \subseteq t_{\mathbf{wp}}$ then f is sound for $\approx$ and $f(\approx) \subseteq \approx$.

The identity function id and the function $\mathbf{p}$ itself are below t. The fact that function composition preserves compatibility is reflected by the idempotence of t, i.e. $t \circ t = t$. Since the companion is idempotent and contains all compatible functions, every bisimulation proof up to a certain combination of compatible functions can be presented as a bisimulation up to the companion. Although this observation does not make such proofs fundamentally easier, it slightly simplifies their presentation: the precise combination of up-to techniques does not have to be made explicit. This is typically extremely convenient in a proof assistant.

2.2. **Tools for validating up-to techniques.** The companion makes it possible to perform bisimulation proofs up to arbitrary combinations of functions that are known to be below it. In concrete languages, we thus have to prove that the functions associated to up-to techniques such as up-to-context, are indeed below the companion. By definition of the companion, given a function f, an obvious way to prove $f \subseteq t$ consists in proving that f is compatible, i.e.[1], $f \rightsquigarrow f$. This is however quite restrictive in practice, because many useful functions are not compatible by themselves, they are only contained in a compatible function, which is often hard to express explicitly. (Very much like bisimulation up-to, which can be small and convenient to work with while the concrete bisimulations lying over them can be large or hard to express.)

Seeing the companion as a coinductive object, one can in fact relax the requirement $f \rightsquigarrow f$ "f is compatible" into $f \rightsquigarrow F(f)$ "f is compatible up to F", where F is a *second order* technique [27]. For instance,

(1) if $f \rightsquigarrow (f \cup g)$ for some $g \subseteq t$, then $h \triangleq f \cup t$ is compatible so that $f \subseteq t$. This means we can freely exploit a function g already known to be below the companion when establishing a progression about f.
(2) if $f \rightsquigarrow f^2$, then f^ω is compatible and contains f. This means we can use f twice in a row when establishing a progression about f. By a similar argument, $f \rightsquigarrow f^\omega$ also entails $f \subseteq t$, meaning we can actually use f as many times as required.
(3) for all sets F of functions such that for all $f \in F$, $f \rightsquigarrow (\bigcup F \cup t)^\omega$, then $(\bigcup F \cup t)^{(\omega^\omega)}$ is compatible (where $h^{(\omega^\omega)}$ is $h \cup h^\omega \cup (h^\omega)^\omega \cup ((h^\omega)^\omega)^\omega \cup \ldots$), so that all functions in F are below the companion. This intuitively makes it possible to reason by 'mutual coinduction' in order to prove that a family of up-techniques is valid.

Leaving the companion aside, the last item above was in fact named "compatibility up-to" in the previous version of this work [21]. This idea was simplified in [28], by defining the second-order function $B(g) \triangleq \bigcup_{f \rightsquigarrow g} f$. Indeed, the notation $f \overset{\mathbf{p}}{\rightsquigarrow} g$, which was an apparent

[1]The notation $\rightsquigarrow$ stands for $\overset{\mathbf{p}}{\rightsquigarrow}$ (on functions) and $\underset{\mathbf{p}}{\rightsquigarrow}$ (on relations); this overloading is explained below.

overloading of the progression operator $\rightsquigarrow$, can now be seen as the regular progression operator associated to the function B.

This function B also has a companion written T, which is a monotone function satisfying the following properties, for all monotone functions f:

- if $f \rightsquigarrow T(f)$ then $f \subseteq t$;
- $f \subseteq T(f)$, $t \subseteq T(f)$, and $T(f) = T(f)^2 = T(f)^\omega$.

The first point is just the fact that every function compatible up to T lies below the companion (like every bisimulation up to t is contained in bisimilarity). The second point tells us that given a family F of functions, $T(\cup F)$ actually contains all potential combinations of functions in F and functions below t.

The three examples of compatible functions up-to listed above can thus be seen as particular instances of compatible functions up to T. In particular, the last item, which we will use repeatedly to prove that up-to-context techniques are valid in the first-order LTSs we present, can be generalised as follows:

(3') for all sets F of functions such that for all $f \in F$, $f \rightsquigarrow T(\bigcup F)$, every function in F is below t.

Remark 7 (On respectfulness). In the first modular treatment of up-to techniques for bisimilarity [33, 36], the notion of *respectful* function was used: a monotone function f is respectful if for all $\mathcal{R}, \mathcal{S}$ such that $\mathcal{R} \subseteq \mathcal{S}$ and $\mathcal{R} \rightsquigarrow \mathcal{S}$, we have $f(\mathcal{R}) \rightsquigarrow f(\mathcal{S})$. Every compatible function is respectful, but the converse is not true. The hypothesis $\mathcal{R} \subseteq \mathcal{S}$ was actually added in the definition of respectfulness to ease proofs about up-to-context, which typically lead to respectful functions that are not compatible. However, this difference between compatible and respectful functions disappears when considering the companion: the largest compatible function and the largest respectful function coincide [28], so that focusing on the simpler notion of compatibility does not prevent us from using certain up-to techniques, in the end.

In practice, proofs of up-to techniques based on respectfulness can be adapted to the compatibility setting as follows. Suppose we try to prove $f \rightsquigarrow T(f)$ for a specific function f, i.e., to prove that $\mathcal{R} \rightsquigarrow \mathcal{S}$ entails $f(\mathcal{R}) \rightsquigarrow T(f)(\mathcal{S})$. The missing assumption $\mathcal{R} \subseteq \mathcal{S}$ is in general useful for those cases where we obtain process derivatives related via $\mathcal{R}$ rather than $\mathcal{S}$. Respectfulness makes it possible to conclude directly in those cases, since

$$\mathcal{R} \subseteq \mathcal{S} \subseteq t(\mathcal{S}) \subseteq T(f)(\mathcal{S}) \ .$$

With compatibility, we can use the up-to-unfolding technique: we have

$$\mathcal{R} \subseteq \mathbf{p}(\mathcal{S}) \subseteq t(\mathcal{S}) \subseteq T(f)(\mathcal{S}) \ ,$$

where the first inclusion is just the assumption $\mathcal{R} \rightsquigarrow \mathcal{S}$.

3. The π-calculus

We let letters a, b range over a set of *names*. We recall the syntax for π-calculus processes (P) and transition labels (μ):

$$P ::= 0 \mid a(b).P \mid \overline{a}b.P \mid P|P \mid \boldsymbol{\nu} b\, P \mid !P$$
$$\mu ::= \tau \mid ab \mid \overline{a}b \mid \overline{a}(b)$$

The name b is bound is P in constructs $a(b).P$ and $\boldsymbol{\nu} b\, P$. The early operational semantics is described by the rules for $\longmapsto_\pi$ in Figure 1. We write $\mathrm{fn}(Q)$ for the free names in Q,

defined as usual. The names $\mathrm{n}(\mu)$ of μ are defined as $\mathrm{n}(\overline{a}b) \triangleq \mathrm{n}(ab) \triangleq \mathrm{n}(\overline{a}(b)) \triangleq \{a, b\}$ and $\mathrm{n}(\tau) \triangleq \emptyset$ and the bound names $\mathrm{bn}(\mu)$ of μ are defined as $\mathrm{bn}(\overline{a}b) \triangleq \mathrm{bn}(ab) \triangleq \mathrm{bn}(\tau) \triangleq \emptyset$ and $\mathrm{bn}(\overline{a}(b)) \triangleq \{b\}$.

$$\text{OUT}\quad \frac{}{\overline{a}b.P \overset{\overline{a}b}{\longmapsto}_\pi P} \qquad \text{INP}\quad \frac{}{a(b).P \overset{ac}{\longmapsto}_\pi P\{c/b\}}$$

$$\text{OPEN}\quad \frac{P \overset{\overline{a}b}{\longmapsto}_\pi P'}{\boldsymbol{\nu} b\, P \overset{\overline{a}(c)}{\longmapsto}_\pi P'\{c/b\}}\ a \neq b, c \notin \mathrm{fn}(\boldsymbol{\nu} b\, P) \qquad \text{RES}\quad \frac{P \overset{\mu}{\longmapsto}_\pi P'}{\boldsymbol{\nu} a\, P \overset{\mu}{\longmapsto}_\pi \boldsymbol{\nu} a\, P'}\ a \notin \mathrm{n}(\mu)$$

$$\text{COMM-L}\quad \frac{P \overset{\overline{a}b}{\longmapsto}_\pi P' \qquad Q \overset{ab}{\longmapsto}_\pi Q'}{P \mid Q \overset{\tau}{\longmapsto}_\pi P' \mid Q'} \qquad \text{CLOSE-L}\quad \frac{P \overset{\overline{a}(b)}{\longmapsto}_\pi P' \qquad Q \overset{ab}{\longmapsto}_\pi Q'}{P \mid Q \overset{\tau}{\longmapsto}_\pi \boldsymbol{\nu} b\, (P' \mid Q')}\ b \notin \mathrm{fn}(Q)$$

$$\text{SUM-L}\quad \frac{P \overset{\mu}{\longmapsto}_\pi P'}{P + Q \overset{\mu}{\longmapsto}_\pi P'} \qquad \text{PAR-L}\quad \frac{P \overset{\mu}{\longmapsto}_\pi P'}{P \mid Q \overset{\mu}{\longmapsto}_\pi P' \mid Q}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset \qquad \text{REP}\quad \frac{P \mid !P \overset{\mu}{\longmapsto}_\pi P'}{!P \overset{\mu}{\longmapsto}_\pi P'}$$

Figure 1: Operational semantics of the π-calculus
(symmetric -R versions of -L rules are omitted)

Note that the conclusion of the OPEN rule is instead $\boldsymbol{\nu} b P \overset{\overline{a}(b)}{\longmapsto}_\pi P'$ in some presentations of the π-calculus. Those presentations look simpler but rely on α-conversion of b in $\boldsymbol{\nu} b\, P$. We choose here to be more explicit.

We do not want to distinguish processes according to the identity of the bound names they may extrude. This is why we need a specific clause for bound outputs in the standard definition of bisimulation:

Definition 8. A relation $\mathcal{R}$ is a strong early bisimulation if, whenever $P\ \mathcal{R}\ Q$:
(1) if $P \overset{\overline{a}(b)}{\longmapsto}_\pi P'$ and $b \notin \mathrm{fn}(Q)$ then $Q \overset{\overline{a}(b)}{\longmapsto}_\pi Q'$ for some Q' such that $P'\ \mathcal{R}\ Q'$,
(2) if $P \overset{\mu}{\longmapsto}_\pi P'$ and μ is not a bound output, then $Q \overset{\mu}{\longmapsto}_\pi Q'$ for some Q' such that $P'\ \mathcal{R}\ Q'$,
(3) the converse of (1) and (2), on Q.

Early bisimilarity, $\sim^{\mathrm{e}}$, is the union of all early bisimulations. The weak version of early bisimilarity, *weak early bisimilarity*, written $\approx^{\mathrm{e}}$, is obtained in the standard way: the transition $Q \overset{\overline{a}(b)}{\longmapsto}_\pi Q'$ in clause (1) is replaced by $Q \overset{\overline{a}(b)}{\Longmapsto}_\pi Q'$; and similarly the transition $Q \overset{\mu}{\longmapsto}_\pi Q'$ in (2) is replaced by $Q \overset{\hat{\mu}}{\Longmapsto}_\pi Q'$. The $\overset{\cdot}{\Longmapsto}_\pi$ transitions are defined from $\overset{\cdot}{\longmapsto}_\pi$ the same way the $\overset{\cdot}{\Longrightarrow}$ transitions were from $\overset{\cdot}{\longrightarrow}$.

When translating the π-calculus semantics to a first-order one, the ad-hoc condition $b \notin \mathrm{fn}(Q)$ has to be removed. To this end, one has to force an agreement between two bisimilar processes on the choice of the bound names appearing in transitions. We obtain this by considering *named processes* (c, P) in which c is bigger or equal to all names in P.

For this to make sense we assume an enumeration of the names and use $\leq$ as the underlying order, and $c+1$ for name following c in the enumeration; for a set of names N, we also write $c \geq N$ to mean $c \geq a$ for all $a \in N$.

The rules below define the translation of the π-calculus transition system to a first-order LTS. In the first-order LTS, the grammar for labels is the same as that of the original LTS; however, for a named process (c, P) the only name that may be exported in a bound output is $c+1$; similarly only names that are below or equal to $c+1$ may be imported in an input transition. (Indeed, testing for all fresh names $b > c$ is unnecessary, doing it only for one $(b = c+1)$ is enough.) This makes it possible to use the ordinary definition of bisimilarity for first-order LTSs, and thus recover the early bisimilarity on the source terms.

$$\frac{P \overset{\tau}{\longmapsto}_\pi P'}{(c,P) \xrightarrow{\tau} (c,P')} \qquad \frac{P \overset{ab}{\longmapsto}_\pi P'}{(c,P) \xrightarrow{ab} (c,P')}\ b \leq c \qquad \frac{P \overset{\overline{a}b}{\longmapsto}_\pi P'}{(c,P) \xrightarrow{\overline{a}b} (c,P')}\ b \leq c$$

$$\frac{P \overset{ab}{\longmapsto}_\pi P'}{(c,P) \xrightarrow{ab} (b,P')}\ b = c+1 \qquad \frac{P \overset{\overline{a}(b)}{\longmapsto}_\pi P'}{(c,P) \xrightarrow{\overline{a}(b)} (b,P')}\ b = c+1$$

We write π^1 for the first-order LTS derived from the above translation of the π-calculus. Although the labels of the source and target transitions have a similar shape, the LTS in π^1 is first-order because labels are taken as purely syntactic, uninterpreted objects. We can also define π^1 using the following two rules:

$$\frac{P \overset{\mu}{\longmapsto}_\pi P' \qquad \mathrm{n}(\mu) \leq c \qquad \mathrm{bn}(\mu) = \emptyset}{(c,P) \xrightarrow{\mu} (c,P')} \qquad \frac{P \overset{\mu}{\longmapsto}_\pi P' \qquad c+1 \in \mathrm{n}(\mu)}{(c,P) \xrightarrow{\mu} (c+1,P')}$$

This characterisation is less explicit but sometimes more convenient in proofs, and it might give an insight on to how to derive translations for other name-based calculi by keeping track of new names and of binding labels.

We will show that the standard notions strong and weak early bisimilarity for the π-calculus ($\sim^{\mathrm{e}}$ and $\approx^{\mathrm{e}}$ from Definition 8) correspond to $\sim$ and $\approx$ in π^1. Proving soundness, i.e., bisimilarity in π^1 entails bisimilarity in π, requires us to establish first that bisimilarity in π^1 is stable under injective substitutions. Anticipating that we also want to propose various up-to techniques for π^1, we show directly that the corresponding up-to-injective-substitutions technique is below the companion. It follows that bisimilarity in π^1 is stable under injective substitutions by Lemma 6, and the work is done only once.

We define the following monotone functions on relations on π^1 processes:

$$\begin{array}{rll}
\mathsf{isub}(\mathcal{R}) & \triangleq \{((d,P\sigma),(d,Q\sigma)) & \text{s.t. } (c,P)\ \mathcal{R}\ (c,Q),\ \sigma \text{ injective on } \mathrm{fn}(P)\cup\mathrm{fn}(Q), \\
 & & \qquad \text{and } \mathrm{fn}(P\sigma)\cup\mathrm{fn}(Q\sigma) \leq d\} \\
\mathsf{bsub}(\mathcal{R}) & \triangleq \{((c,P\sigma),(c,Q\sigma)) & \text{s.t. } (c,P)\ \mathcal{R}\ (c,Q),\ \sigma \text{ bijective on } \{1\dots c\}\} \\
\mathsf{str}(\mathcal{R}) & \triangleq \{((d,P),(d,Q)) & \text{s.t. } (c,P)\ \mathcal{R}\ (c,Q) \text{ and } \mathrm{fn}(P,Q) \leq d\} \\
\mathsf{w}(\mathcal{R}) & \triangleq \{((c+k,P),(c+k,Q)) & \text{s.t. } (c,P)\ \mathcal{R}\ (c,Q)\ ,\ k \in \mathbb{N}\}
\end{array}$$

The first one, isub, makes it possible to use injective substitutions; the second one, bsub, is restricted to bijective substitutions; the third one, str, is a form of *strengthening*, making it possible to readjust the bound c on free names; conversely, the last one, w, is a form of *weakening*. The last two functions are often useful as up-to techniques, by themselves. The

point of the function bsub is that it makes it possible to obtain isub as a derived technique: we have $\mathsf{isub} = \mathsf{bsub} \circ \mathsf{w}$, and bsub is slightly easier to analyse.

Lemma 9. The functions isub, bsub, str, and w are all below the companion $t_{\mathbf{sp}}$ and below the companion $t_{\mathbf{wp}}$.

Proof. We first show that bsub is compatible, i.e., $\mathsf{bsub} \rightsquigarrow \mathsf{bsub}$. Let $\mathcal{R}, \mathcal{S}$ be two relations such that $\mathcal{R} \rightsquigarrow \mathcal{S}$, and let us prove $\mathsf{bsub}(\mathcal{R}) \rightsquigarrow \mathsf{bsub}(\mathcal{S})$. For this, let $(P\sigma, Q\sigma) \in \mathsf{bsub}(\mathcal{R})$ for some P, Q such that $(c, P)\ \mathcal{R}\ (c, Q)$ with σ some bijective substitution on $\{1 \dots c\}$. From a transition $(c, P\sigma) \xrightarrow{\mu} (c', P')$, because $P\sigma\sigma^{-1} = P$, we can transform it to $(c, P) \xrightarrow{\mu\sigma^{-1}} (c', P'\sigma^{-1})$. Then, thanks to $\mathcal{R} \rightsquigarrow \mathcal{S}$ with the $\mu\sigma^{-1}$ transition, there exists Q' such that $(c, Q) \xrightarrow{\mu\sigma^{-1}} (c', Q')$ and $(c', P'\sigma^{-1})\ \mathcal{S}\ (c', Q')$, which imply respectively $(c, Q\sigma) \xrightarrow{\mu} (c', Q'\sigma)$ and $(c', P'\sigma^{-1}\sigma) = (c', P')\ \mathsf{bsub}(\mathcal{S})\ (c', Q'\sigma)$. The argument used for $\mathbf{sp}$ can also be used for $\mathbf{wp}$. Thus bsub is below t.

Then we show $\mathsf{w} \rightsquigarrow \mathsf{bsub} \circ \mathsf{w}$ and $\mathsf{str} \rightsquigarrow \mathsf{str} \circ \mathsf{bsub}$, which is done using a similar diagram-chasing argument. Each newly created name is handled with a transposition using bsub, using the facts that $Q \overset{\mu}{\longmapsto}_\pi Q'$ implies $\mathrm{fn}(Q') \subseteq \mathrm{fn}(Q) \cup \mathrm{n}(\mu)$, and that $\mathrm{fn}(Q'\sigma) = \sigma(\mathrm{fn}(Q'))$.

Since $\mathsf{bsub} \subseteq t$, we deduce $\mathsf{w} \rightsquigarrow T(\mathsf{w})$ and $\mathsf{str} \rightsquigarrow T(\mathsf{str})$, so that both w and str are also below t. It follows that $\mathsf{isub} = \mathsf{bsub} \circ \mathsf{w} \subseteq t \circ t = t$. □

It follows by Lemma 6 that bisimilarities in π^1 are closed under injective substitution: $\mathsf{isub}(\sim) \subseteq \sim$ and $\mathsf{isub}(\approx) \subseteq \approx$. We can now establish full abstraction between π^1 and early bisimilarities:

Theorem 10. Assume $c \geq \mathrm{fn}(P) \cup \mathrm{fn}(Q)$. Then we have: $P \sim^{\mathrm{e}} Q$ iff $(c, P) \sim (c, Q)$, and $P \approx^{\mathrm{e}} Q$ iff $(c, P) \approx (c, Q)$.

Proof. We prove the case of weak bisimilarity, the strong case being easier. For the direct implication, we show that the relation $\mathcal{R}_1$ defined below is a weak bisimulation:

$$\mathcal{R}_1 \triangleq \{((c, P), (c, Q)) \mid P \approx^{\mathrm{e}} Q \ \wedge\ c \geq \mathrm{fn}(P) \cup \mathrm{fn}(Q)\}$$

The only interesting transition is when $(c, P) \xrightarrow{\overline{a}(b)} (d, P')$ with $d = b = c + 1$. Since $c \geq \mathrm{fn}(P) \cup \mathrm{fn}(Q)$, we know that $b \notin \mathrm{fn}(Q)$ so $P \approx^{\mathrm{e}} Q$ tells us that $Q \overset{\overline{a}(b)}{\Longmapsto}_\pi Q'$ with $P' \approx^{\mathrm{e}} Q'$. Repeatedly applying the rules defining $\dot{\longrightarrow}$, since $b = c + 1$, yields

$$\frac{Q \Longmapsto_\pi Q_1 \overset{\overline{a}(b)}{\longmapsto}_\pi Q_2 \Longmapsto_\pi Q'}{(c, Q) \Longrightarrow (c, Q_1) \xrightarrow{\overline{a}(b)} (b, Q_2) \Longrightarrow (b, Q')}$$

and, indeed, $b \geq \mathrm{fn}(P') \cup \mathrm{fn}(Q')$.

For the converse, proving that

$$\mathcal{R}_2 \triangleq \{(P, Q) \mid \exists c \geq \mathrm{fn}(P) \cup \mathrm{fn}(Q)\ (c, P) \approx (c, Q)\}$$

is a weak early bisimulation needs a little more care, since fresh names in labels can be other than $c + 1$ (they can be less than or greater than $c + 1$). Suppose $P\ \mathcal{R}_2\ Q$, which means there is $c \geq \mathrm{fn}(P) \cup \mathrm{fn}(Q)$ such that $(c, P) \approx (c, Q)$. We analyse the transitions of the form $P \overset{\mu}{\longmapsto}_\pi P'$:

(1) if $\mu = \overline{a}(b)$ or $\mu = ab$ and $b \geq c+2$ then we have $P \overset{\mu}{\longmapsto}_\pi P'$ if and only if $P \overset{\mu\{b'/b\}}{\longmapsto}_\pi P'\{b'/b\}$ with $b' = c+1$. Exploiting $\approx$ to get $(c,Q) \overset{\overline{a}(b')}{\Longrightarrow} (b',Q')$ with $(b',P'\{b'/b\}) \approx (b',Q')$ and hence, since $\mathsf{isub}(\approx) \subseteq \approx$ by Lemmas 6 and 9, we obtain $(b,P') \approx (b,Q'\{b/b'\})$ and hence $P' \mathcal{R}_2 Q'\{b/b'\}$. We conclude since $Q \overset{\mu\{b'/b\}}{\Longmapsto}_\pi Q'$ implies $Q \overset{\mu}{\Longmapsto}_\pi Q'\{b/b'\}$.
(2) If $\mu = ab$ or $\mu = \overline{a}(b)$ with $b = c+1$ then $P \overset{\mu}{\longmapsto}_\pi P'$ implies $(c,P) \xrightarrow{\mu} (b,P')$, which implies $(c,Q) \overset{\mu}{\Longrightarrow} (b,Q')$, and then $Q \overset{\mu}{\Longmapsto}_\pi Q'$ with $(b,P') \approx (b,Q')$.
(3) If $\mathrm{n}(\mu) \leq c$, there are two cases:
   (a) μ is not a bound output. Then $P \overset{\mu}{\longmapsto}_\pi P'$ if and only if $(c,P) \xrightarrow{\mu} (c,P')$ when $c \geq \mathrm{fn}(P)$; thus we can derive $Q \overset{\mu}{\Longmapsto}_\pi Q'$ and $P' \mathcal{R}_2 Q'$.
   (b) $\mu = \overline{a}(b)$ (hence $b \notin \mathrm{fn}(P)$) with the additional information that $b \notin \mathrm{fn}(Q)$. Then $P \overset{\overline{a}(b')}{\longmapsto}_\pi P'\{b'/b\}$ with $b' = c+1$. We get $(c,P) \xrightarrow{\overline{a}(b')} (b',P'\{b'/b\})$ and then $(c,Q) \overset{\overline{a}(b')}{\Longrightarrow} (b',Q')$ and since $b \notin \mathrm{fn}(Q)$, we also have $Q \overset{\overline{a}(b)}{\Longmapsto}_\pi Q'\{b/b'\}$. The progression also gives us $(b',P'\{b'/b\}) \approx (b',Q')$. By closure of $\approx$ under injective substitutions (isub), we deduce $(b,(P'\{b'/b\})\{b/b'\}) = (b,P') \approx (b,Q'\{b/b'\})$ and finally $P' \mathcal{R}_2 Q'\{b/b'\}$. □

The above full abstraction result allows us to import the theory of up-to techniques for first-order LTSs and bisimilarity, in both the strong and the weak cases.

We have already proved the validity of preliminary up-to techniques that are specific to π^1 (Lemma 9); we proceed below with up-to-context techniques.

The up-to-context function is decomposed into a set of smaller context functions, called *initial* [26], one for each operator of the π-calculus. The only exception to this is the input prefix, since early bisimilarity in the π-calculus is not preserved by this operator. We write $\mathcal{C}_o$, $\mathcal{C}_\nu$, $\mathcal{C}_!$, $\mathcal{C}_|$, and $\mathcal{C}_+$ for these initial context functions, respectively applying the operators of output prefix, restriction, replication, parallel composition, and sum, to all pairs in the given relation.

Definition 11. We define the functions $\mathcal{C}_o$, $\mathcal{C}_\nu$, $\mathcal{C}_!$, $\mathcal{C}_|$ and $\mathcal{C}_+$ on relations on π^1 by the following rules:

$$\frac{(c,P)\ \mathcal{R}\ (c,Q) \qquad c \geq a,b}{(c,\overline{a}b.P)\ \mathcal{C}_o(\mathcal{R})\ (c,\overline{a}b.Q)} \qquad \frac{(c,P)\ \mathcal{R}\ (c,Q)}{(c,(\nu a)P)\ \mathcal{C}_\nu(\mathcal{R})\ (c,(\nu a)Q)} \qquad \frac{(c,P)\ \mathcal{R}\ (c,Q)}{(c,!P)\ \mathcal{C}_!(\mathcal{R})\ (c,!Q)}$$

$$\frac{(c,P_1)\ \mathcal{R}\ (c,Q_1) \qquad (c,P_2)\ \mathcal{R}\ (c,Q_2)}{(c,P_1 \mid P_2)\ \mathcal{C}_|(\mathcal{R})\ (c,Q_1 \mid Q_2)} \qquad \frac{(c,P_1)\ \mathcal{R}\ (c,Q_1) \qquad (c,P_2)\ \mathcal{R}\ (c,Q_2)}{(c,P_1+P_2)\ \mathcal{C}_+(\mathcal{R})\ (c,Q_1+Q_2)}$$

While bisimilarity in the π-calculus is not preserved by input prefix, a weaker rule holds:

$$\frac{\forall c,\ \ P\{c/b\} \asymp Q\{c/b\}}{a(b).P \asymp a(b).Q} \tag{3.1}$$

where $\asymp$ can be $\sim^{\mathrm{e}}$ or $\approx^{\mathrm{e}}$. We define accordingly $\mathcal{C}_i$, the function for input prefix:

Definition 12. $\mathcal{C}_i$ is the function on π^1 relations defined by the rule:

$$\frac{a \leq d \qquad \forall c \leq d+1\ (d+1,P\{c/b\})\ \mathcal{R}\ (d+1,Q\{c/b\})}{(d,a(b).P)\ \mathcal{C}_i(\mathcal{R})\ (d,a(b).Q))}$$

Theorem 13. The functions $\mathcal{C}_o, \mathcal{C}_i, \mathcal{C}_\nu, \mathcal{C}_!, \mathcal{C}_|, \mathcal{C}_+$ are all below $t_{\mathbf{sp}}$.

Proof. Let $F \triangleq \{\mathcal{C}_o, \mathcal{C}_i, \mathcal{C}_\nu, \mathcal{C}_!, \mathcal{C}_|, \mathcal{C}_+\}$, we prove $f \rightsquigarrow T(\cup F)$ for each function $f \in F$. In each case, we assume $\mathcal{R} \rightsquigarrow \mathcal{S}$ and we prove $f(\mathcal{R}) \rightsquigarrow T(\cup F)(\mathcal{S})$. Remark that $\mathcal{R} \subseteq \mathbf{sp}(\mathcal{S})$, and so $T(\cup F)(\mathcal{R} \cup \mathcal{S}) \subseteq T(\cup F)(\mathbf{sp}(\mathcal{S}) \cup \mathcal{S}) \subseteq T(\cup F)(T(\cup F)(\mathcal{S})) = T(\cup F)(\mathcal{S})$, and so it is enough, and more convenient, to establish $f(\mathcal{R}) \rightsquigarrow T(\cup F)(\mathcal{R} \cup \mathcal{S})$.

For this, it suffices to analyse the transitions emerging from the left-hand side of $f(\mathcal{R})$, as every f is symmetric:

- $\mathcal{C}_o(\mathcal{R}) \rightsquigarrow \mathcal{R}$ is immediate
- $\mathcal{C}_+(\mathcal{R}) \rightsquigarrow \mathcal{S}$ is also straightforward;
- $\mathcal{C}_i(\mathcal{R}) \rightsquigarrow \mathsf{str}(\mathcal{R})$: assume $(d, a(b).P)\ \mathcal{C}_i(\mathcal{R})\ (d, a(b).Q)$; each transition is of the form $\xrightarrow{ac}$, yielding a pair (p, q) where $p = (d', P\{c/b\})$ and $q = (d', Q\{c/b\})$;
  - if $d' = c + 1$ then $(p, q) \in \mathcal{R}$ by definition of $\mathcal{C}_i$.
  - if $d' = c$ then $(d + 1, P\{c/b\})\ \mathcal{R}\ (d + 1, Q\{c/b\})$ by definition of $\mathcal{C}_i$, and hence $(p, q) \in \mathsf{str}(\mathcal{R})$.
- $\mathcal{C}_\nu(\mathcal{R}) \rightsquigarrow \mathcal{C}_\nu(\mathcal{S}) \cup \mathsf{isub}(\mathcal{S})$
  The interesting case arises for transitions for which the last rule applied is the extrusion rule: $(c, (\nu d)P) \xrightarrow{\overline{a}(b)} (b, P'\{b/d\})$ with $b = c + 1$ and $P \overset{\overline{a}d}{\longmapsto}_\pi P'$. The problem is to relate $(b, P'\{b/d\})$ to $(b, Q'\{b/d\})$ knowing that $(c, P')\ \mathcal{S}\ (c, Q')$. This is done using the isub function with the injective substitution $\{b/d\} : \{1 \dots c\} \to \{1 \dots b\}$.
- $\mathcal{C}_|(\mathcal{R}) \rightsquigarrow N(\mathcal{C}_|(\mathsf{isub}(\mathcal{R} \cup \mathcal{S})))$, where $N \triangleq (\mathsf{str} \circ \mathcal{C}_\nu) \cup \mathsf{id}$. For this, we analyse the transition $(c, P_1 \mid P_2) \xrightarrow{\mu} (c', P')$. First, let's assume that $c' = c$. The transition must come from one of the four rules PAR-L, PAR-R, COMM-L, or COMM-R:
  - Rule PAR-L results in $(c, P_1' \mid P_2)$ with $P_1 \overset{\mu_1}{\longmapsto}_\pi P_1'$, so we obtain, from $(c, P_1)\ \mathcal{R}\ (c, Q_1)$ and $\mathcal{R} \rightsquigarrow \mathcal{S}$, some Q_1' such that $Q_1 \overset{\mu_1}{\longmapsto}_\pi Q_1'$. Finally we obtain the pair $((c, P_1' \mid P_2), (c, Q_1' \mid Q_2))$ which belongs to $\mathcal{C}_|(\mathcal{R} \cup \mathcal{S})$.
  - Symmetrically, rule PAR-R takes us to the pair $((c, P_1 \mid P_2'), (c, Q_1 \mid Q_2')) \in \mathcal{C}_|(\mathcal{R} \cup \mathcal{S})$.
  - Working both sides, rules COMM-L and COMM-R both lead us to a pair
    $$((c, P_1' \mid P_2'), (c, Q_1' \mid Q_2')) \in \mathcal{C}_|(\mathcal{S}).$$

  The second case is when $c' = c + 1$. This means that the transition is derived using PAR-L, PAR-R, CLOSE-L, or CLOSE-R. We consider two cases:
  - The last rule is a PAR-L rule (PAR-R being symmetric), with a label of the form $\overline{a}(b)$. We know $(c, P_1)\ \mathcal{R}\ (c, Q_1)$ and $(c, P_2)\ \mathcal{R}\ (c, Q_2)$ and
    $$\frac{P_1 \overset{\overline{a}(b)}{\longmapsto}_\pi P_1'}{P_1 \mid P_2 \overset{\overline{a}(b)}{\longmapsto}_\pi P_1' \mid P_2}\ b \notin \mathrm{fn}(P_2)$$
    We have $b = c + 1$, following the rule for bound output. We also have the following reductions in π^1, from (c, P_1), and then from (c, Q_1) using the progression $\mathcal{R} \rightsquigarrow \mathcal{S}$:
    $$(c, P_1) \xrightarrow{\overline{a}(b)} (b, P_1') \qquad\qquad (c, Q_1) \xrightarrow{\overline{a}(b)} (b, Q_1')$$

With $(b, P_1') \; \mathcal{S} \; (b, Q_1')$. We now need to relate the resulting processes:

$$\dfrac{\dfrac{(b,P_1') \; \mathcal{S} \; (b,Q_1')}{(b,P_1') \; \mathcal{R} \cup \mathcal{S} \; (b,Q_1')}}{\dfrac{(b,P_1') \; \mathsf{isub}(\mathcal{R} \cup \mathcal{S}) \; (b,Q_1')}{(b,P_1' \mid P_2) \; (\mathcal{C}_| \circ \mathsf{isub})(\mathcal{R} \cup \mathcal{S}) \; (b, Q_1' \mid Q_2)}} \quad \dfrac{\dfrac{(c,P_2) \; \mathcal{R} \; (c,Q_2)}{(c,P_2) \; \mathcal{R} \cup \mathcal{S} \; (c,Q_2)}}{(b,P_2) \; \mathsf{isub}(\mathcal{R} \cup \mathcal{S}) \; (b,Q_2)}} \; .$$

The same happens for the input transition $\xrightarrow{ab}$ when $b = c + 1$.

– The last rule is a CLOSE rule: we know $(c, P_1) \; \mathcal{R} \; (c, Q_1)$ and $(c, P_2) \; \mathcal{R} \; (c, Q_2)$ and

$$\dfrac{P_1 \overset{\overline{a}(b)}{\longmapsto}_\pi P_1' \qquad P_2 \overset{ab}{\longmapsto}_\pi P_2'}{P_1 \mid P_2 \overset{\tau}{\longmapsto}_\pi (\boldsymbol{\nu} b)(P_1' \mid P_2')} \; b \notin \mathrm{fn}(P_2)$$

We can assume $b = c + 1$ as b is fresh on both sides. The two hypotheses can then be transformed into transitions in π_1:

$$(c, P_1) \xrightarrow{\overline{a}(b)} (b, P_1') \qquad\qquad (c, P_2) \xrightarrow{ab} (b, P_2') \; .$$

We have the same transitions for Q_1 and Q_2, respectively. Using the hypothesis $\mathcal{R} \rightsquigarrow \mathcal{S}$, we obtain named processes (b, Q_1') and (b, Q_2'), related through $\mathcal{S}$, which we can combine using $\mathcal{C}_|$ and then strengthen b to c since $b \notin \mathrm{fn}(P, Q)$:

$$\dfrac{\dfrac{\dfrac{(b,P_1') \; \mathcal{S} \; (b,Q_1') \qquad (b,P_2') \; \mathcal{S} \; (b,Q_2')}{(b, P_1' \mid P_2') \qquad \mathcal{C}_|(\mathcal{S}) \qquad (b, Q_2' \mid Q_1')}}{(b, (\nu b)(P_1' \mid P_2')) \quad \mathcal{C}_\nu(\mathcal{C}_|(\mathcal{S})) \quad (b, (\nu b)(Q_2' \mid Q_1'))}}{(c, (\nu b)(P_1' \mid P_2')) \; \mathsf{str}(\mathcal{C}_\nu(\mathcal{C}_|(\mathcal{S}))) \; (c, (\nu b)(Q_2' \mid Q_1'))} \; .$$

- $\mathcal{C}_!(\mathcal{R}) \rightsquigarrow \mathcal{C}_|^\omega(N(\mathcal{C}_|^\omega((\mathcal{C}_! \cup \mathsf{id})(\mathsf{isub}(\mathcal{R} \cup \mathcal{S})))))$

  We analyse the transition $(c, !P) \xrightarrow{\mu} (c', P')$. If $!P \overset{\mu}{\longmapsto}_\pi P'$ then one of the following holds:
  (1) $P' = !P \,|\, P_0 \,|\, P \,|\, \ldots \,|\, P$ with $P \overset{\mu}{\longmapsto}_\pi P_0$, or
  (2) $\mu = \tau$ and $P' = !P \,|\, P_0 \,|\, P \,|\, \ldots \,|\, P \,|\, P_1 \,|\, P \,|\, \ldots \,|\, P$ with $P \overset{\overline{a}b}{\longmapsto}_\pi P_i$ and $P \overset{ab}{\longmapsto}_\pi P_{1-i}$, or
  (3) $\mu = \tau$ and $P' = (\nu b)(!P \,|\, P_0 \,|\, P \,|\, \ldots \,|\, P \,|\, P_1) \,|\, P \,|\, \ldots \,|\, P$ with $P \overset{\overline{a}(b)}{\longmapsto}_\pi P_i$ and $P \overset{ab}{\longmapsto}_\pi P_{1-i}$.
  In case 1 we have $(c, P) \xrightarrow{\mu} (b, P_0)$ with $b = c' \in \{c, c+1\}$. In case 2 we have $(c, P) \xrightarrow{\mu_i} (b, P_i)$ for each i, with $b = c$. In case 3 we have $(c, P) \xrightarrow{\mu_i} (b, P_i)$ for each i, with $b = c + 1$. In each case, we obtain, since $\mathcal{R} \rightsquigarrow \mathcal{S}$ with $(c, P) \; \mathcal{R} \; (c, Q)$, a transition $(c, !Q) \xrightarrow{\mu} (c', Q')$ with Q' of the same shape, so we only need to relate (c', P') to (c', Q') knowing $(b, P_i) \; \mathcal{S} \; (b, Q_i)$. First, we note that $(b, P) \; \mathsf{isub}(\mathcal{R}) \; (b, Q)$. We have now the following pairs in $\mathcal{S}_0 \triangleq (\mathcal{C}_! \cup \mathsf{id})(\mathsf{isub}(\mathcal{R} \cup \mathcal{S}))$:

  $$(b, !P) \; \mathcal{S}_0 \; (b, !Q) \qquad (b, P_0) \; \mathcal{S}_0 \; (b, Q_0) \qquad (b, P_1) \; \mathcal{S}_0 \; (b, Q_1) \qquad (b, P) \; \mathcal{S}_0 \; (b, Q)$$

  We can then apply $\mathcal{C}_|$ several times to obtain the three pairs (with $\mathcal{S}_1 \triangleq \mathcal{C}_|^\omega(\mathcal{S}_0)$):

  $$\begin{array}{rcl} (b, !P \,|\, P_0 \,|\, P \,|\, \ldots \,|\, P) & \mathcal{S}_1 & (b, !Q \,|\, Q_0 \,|\, Q \,|\, \ldots \,|\, Q) \\ (b, !P \,|\, P_0 \,|\, P \,|\, \ldots \,|\, P \,|\, P_1 \,|\, P \,|\, \ldots \,|\, P) & \mathcal{S}_1 & (b, !Q \,|\, Q_0 \,|\, Q \,|\, \ldots \,|\, Q \,|\, Q_1 \,|\, P \,|\, \ldots \,|\, P) \\ (b, !P \,|\, P_0 \,|\, P \,|\, \ldots \,|\, P \,|\, P_1) & \mathcal{S}_1 & (b, !Q \,|\, Q_0 \,|\, Q \,|\, \ldots \,|\, Q \,|\, Q_1) \end{array}$$

The first two pairs handle cases 1 and 2. For case 3 we need to apply $\mathcal{C}_\nu$ to add $(\nu b)-$ and then str so to go from $(b,(\nu b)-)$ to $(c,(\nu b)-)$. We apply $\mathcal{C}_|^\omega$ again to add the missing $-\,|\,P\,|\ldots|\,P$ and we obtain (c,P') and (c,Q') in the relation $\mathcal{C}_|^\omega(\mathsf{str}(\mathcal{C}_\nu(\mathcal{S}_1)))$. Concluding, we have obtained the following progression:

$$\mathcal{C}_!(\mathcal{R}) \rightsquigarrow \mathcal{S}_1 \cup \mathcal{C}_|^\omega(\mathsf{str}(\mathcal{C}_\nu(\mathcal{S}_1))) \subseteq \mathcal{C}_|^\omega(N(\mathcal{S}_1))$$

which was our original goal. Note that the iterated $\mathcal{C}_|^\omega$ was used twice; both times it can be absorbed by T, so to give us at the end $\mathcal{C}_!(\mathcal{R}) \rightsquigarrow T(\mathcal{C}_| \cup \mathcal{C}_\nu \cup \mathcal{C}_!)(\mathcal{R} \cup \mathcal{S})$.

For each $f \in F$ we have established a progression from $f(\mathcal{R})$ to $T(\cup F)(\mathcal{R} \cup \mathcal{S})$, and so to $T(\cup F)(\mathcal{S})$, as needed: this gives us $f \rightsquigarrow T(\cup F)$, and in turn $\cup F \rightsquigarrow T(\cup F)$ and $\cup F \subseteq t$. □

Weak bisimilarity is not preserved by sums, only by guarded sums, whose function is $\mathcal{C}_{g+} \triangleq \mathcal{C}_+^\omega \circ (\mathcal{C}_o \cup \mathcal{C}_i)$.

Theorem 14. The functions $\mathcal{C}_o, \mathcal{C}_i, \mathcal{C}_\nu, \mathcal{C}_!, \mathcal{C}_|, \mathcal{C}_{g+}$ are below the companion $t_{\mathbf{wp}}$.

Proof. The progressions are as in the proof of Theorem 13, except for $\mathcal{C}_{g+}$, which is treated as $\mathcal{C}_o$ and $\mathcal{C}_i$; we need one more up-to technique for the case of the replication. Assuming $\mathcal{R} \underset{\mathbf{wp}}{\rightsquigarrow} \mathcal{S}$, the following progressions hold:

$$\mathcal{C}_o(\mathcal{R}) \underset{\mathbf{wp}}{\rightsquigarrow} \mathcal{R} \qquad \mathcal{C}_i(\mathcal{R}) \underset{\mathbf{wp}}{\rightsquigarrow} \mathsf{str}(\mathcal{R}) \qquad \mathcal{C}_\nu(\mathcal{R}) \underset{\mathbf{wp}}{\rightsquigarrow} (\mathcal{C}_\nu \cup \mathsf{isub})(\mathcal{S}) \qquad \mathcal{C}_{g+}(\mathcal{R}) \underset{\mathbf{wp}}{\rightsquigarrow} \mathsf{str}(\mathcal{R})$$

$$\mathcal{C}_|(\mathcal{R}) \underset{\mathbf{wp}}{\rightsquigarrow} N(\mathcal{C}_|(\mathsf{isub}(\mathcal{R} \cup \mathcal{S})))$$

$$\mathcal{C}_!(\mathcal{R}) \underset{\mathbf{wp}}{\rightsquigarrow} \mathcal{C}_|^\omega(N(\mathcal{C}_|^\omega((\mathcal{C}_! \cup \mathsf{id})(\mathsf{isub}(\mathcal{R} \cup \mathcal{S}))))) \cup (\mathcal{C}_|^\omega(\mathcal{C}_!(\mathcal{R}) \cup \mathcal{R} \cup \mathcal{S})\sim) \ .$$

Again progressing to $T(\cup F)(\mathcal{R} \cup \mathcal{S})$ is conveniently sufficient since $T(\cup F)(\mathcal{R} \cup \mathcal{S}) \subseteq T(\cup F)(\mathcal{S})$. For the replication operator, only case 1 (of the corresponding proof of Theorem 13) cannot be transported to the weak case. We have:

$$(c,!P) \xrightarrow{\tau} (c,!P\,|\,P_0\,|\,P\,|\ldots|\,P) \qquad \text{with} \qquad (c,P) \xrightarrow{\tau} (c,P_0) \ .$$

We use the property that $\mathcal{R} \rightsquigarrow \mathcal{S}$ so that from $(c,P)\ \mathcal{R}\ (c,Q)$ we obtain $(c,Q) \xrightarrow{\tau}{}^{n} (c,Q_0)$. Then

- if $n > 0$ we have $(c,!Q) \Longrightarrow (c,!Q\,|\,Q_0\,|\ldots|\,Q)$ and we conclude as before.
- if $n = 0$ then there is no transition from Q or $!Q$, we know $P_0\ \mathcal{S}\ Q$ but we cannot reach the desired form $(c,!Q\,|\,Q\,|\ldots|\,Q)$ with a transition. Instead, we remark that $(c,!Q) \sim (c,!Q\,|\,Q\,|\ldots|\,Q)$ and so we simply progress to the relation $\mathcal{S}_2\sim$ where $\mathcal{S}_2 = \mathcal{C}_|^\omega(\mathcal{C}_!(\mathcal{R}) \cup \mathcal{R} \cup \mathcal{S})$.

Compared to the strong case, we only need to compose (on the left) the right-hand side of the progression with the function $\mathcal{R} \mapsto \sim\mathcal{R}\sim$ ('up-to-strong-bisimilarity') which is indeed **wp**-compatible. □

As a byproduct of the compatibility of these initial context functions, and using Lemma 6, we derive the standard congruence properties of strong and weak early bisimilarity, including the rule (3.1) for input prefix.

Corollary 15. In the π-calculus, relations $\sim^{\mathrm{e}}$ and $\approx^{\mathrm{e}}$ are preserved by the operators of output prefix, replication, parallel composition, restriction; $\sim^{\mathrm{e}}$ is also preserved by sum, whereas $\approx^{\mathrm{e}}$ is only preserved by guarded sums. Moreover, rule (3.1), for input prefix, is valid both for $\sim^{\mathrm{e}}$ and $\approx^{\mathrm{e}}$.

Remark 16. Late bisimilarity [36, Section 4.5] makes use of transitions $P \overset{a(b)}{\longmapsto}_\pi P'$ where b is bound, the definition of bisimulation containing a quantification over names. To translate this bisimilarity in a first-order LTS we would need two transitions for the input $a(b)$: one to fire the input a, leaving b uninstantiated (for example, in a new kind of process $(b)(c, P)$ akin to an abstraction), and another to instantiate b with any name, for transitions starting from processes of the new kind:

$$\frac{P \overset{a(b)}{\longmapsto}_\pi P'}{(c, P) \overset{a(-)}{\longrightarrow} (b)(c, P')} \qquad \frac{b' \leq c + 1}{(b)(c, P) \overset{a(b')}{\longrightarrow} (c + 1, P'\{b'/b\})}$$

While such a translation does yield full abstraction for both strong and weak late bisimilarities, the decomposition of an input transition into two steps prevents us from obtaining the compatibility of up-to-context. Indeed, compatibility of up-to-context intuitively requires that the immediate transitions of $C[P]$ should depend only on the immediate transitions of P. However, if inputs are decomposed into two steps, a contexts such as $[\cdot]_1 \mid \overline{a}b$ may combine two successive steps of the (input) argument to perform a single τ transition.

To conclude, the main take-away message on the π-calculus is that it suffices to count names to make the LTS first-order. Then, once the corresponding up-to techniques for names are set-up, we recover the usual progression proofs, in a modular way. While this level of modularity was already present in [25], it now becomes simpler thanks to the companion.

4. Call-by-name λ-calculus

To study the applicability of our approach to higher-order languages, we investigate the pure call-by-name λ-calculus, referred to as ΛN in the sequel.

We use M, N to range over the set Λ of λ-terms, and x, y, z to range over variables. The set Λ of pure λ-terms is defined by:

$$M, N ::= x \mid \lambda x.M \mid MN$$

We assume the familiar concepts of free and bound variables and substitutions, and identify α-convertible terms. The only values are the λ-abstractions $\lambda x.M$. In this section and in the following one, results and definitions are presented on closed terms and we write Λ^0 for the subset of closed terms. Extension to open terms is made using closing abstractions (i.e., abstracting on all free variables). The reduction relation of ΛN is the *call-by-name reduction relation* $\longmapsto_{\mathsf{n}}$, defined as the least relation over Λ^0 that is closed under the following rules.

$$\frac{}{(\lambda x.M)N \longmapsto_{\mathsf{n}} M\{N/x\}} \qquad \frac{M \longmapsto_{\mathsf{n}} M'}{MN \longmapsto_{\mathsf{n}} M'N}$$

We write $\Longmapsto_{\mathsf{n}}$ for its reflexive and transitive closure. In call-by-name, *evaluation contexts* are described by the following grammar:

$$C_e := C_e\ M \mid [\cdot]$$

As reference equivalence for the λ-calculus we consider *environmental bisimilarity* [34, 14], which coincides with contextual equivalence and Abramsky's applicative bisimilarity [2] on pure λ-terms while enabling a richer set of up-to techniques. Environmental bisimilarity makes a clear distinction between the tested terms and the environment. An element of

an environmental bisimulation has, in addition to the tested terms M and N, a further component $\mathcal{E}$, the environment, which expresses the observer's current knowledge. When an input from the observer is required, the arguments supplied are terms that the observer can build using the current knowledge; that is, terms obtained by composing the values in $\mathcal{E}$ using the operators of the calculus. An *environmental relation* is a set of elements, each of which can be of two forms: either a relation $\mathcal{E}$ on closed values, or a triple $(\mathcal{E}, M, N)$ where M, N are closed terms and $\mathcal{E}$ is a relation on closed values. We use $\mathcal{X}, \mathcal{Y}$ to range over environmental relations. In a triple $(\mathcal{E}, M, N)$ the relation component $\mathcal{E}$ is the *environment*, and M, N are the *tested terms*. We write $M\ \mathcal{X}_{\mathcal{E}}\ N$ for $(\mathcal{E}, M, N) \in \mathcal{X}$. We write $\mathcal{E}^\star$ for the closure of $\mathcal{E}$ under contexts. We only define the weak version of the bisimilarity; its strong version is obtained in the expected way.

Definition 17. An environmental relation $\mathcal{X}$ is an *environmental bisimulation* if

(1) $M\ \mathcal{X}_{\mathcal{E}}\ N$ implies:
   (a) if $M \longmapsto_{\mathsf{n}} M'$ then $N \Longmapsto_{\mathsf{n}} N'$ and $M'\ \mathcal{X}_{\mathcal{E}}\ N'$;
   (b) if $M = V$ then $N \Longmapsto_{\mathsf{n}} W$ and $\mathcal{E} \cup \{(V, W)\} \in \mathcal{X}$ (V and W are values);
   (c) the converse of the above two conditions, on N;

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

Environmental bisimilarity, $\approx^{\mathsf{env}}$, is the largest environmental bisimulation.

For environmental bisimilarity to be expressed via a first-order transition system, a few issues have to be resolved. For instance, an environmental bisimilarity contains both triples $(\mathcal{E}, M, N)$, and pure environments $\mathcal{E}$, which shows up in the difference between clauses (1) and (2) of Definition 17. Moreover, the input supplied to tested terms may be constructed using arbitrary contexts.

We write ΛN^1 for the first-order LTS resulting from the translation of ΛN. The states of ΛN^1 are sequences of λ-terms in which only the last one need not be a value. We use Γ and Δ to range over sequences of values only; thus (Γ, M) indicates a sequence of λ-values followed by M. We write $|\Gamma|$ for the length of a sequence Γ, and Γ_i for the i-th element in Γ, when $i \leq |\Gamma|$.

For a finite environment $\mathcal{E}$, we write $\mathcal{E}_1$ for an ordered projection of the pairs in $\mathcal{E}$ on the first component, and $\mathcal{E}_2$ is the corresponding projection on the second component. In the translation, intuitively, a triple $(\mathcal{E}, M, N)$ of an environmental bisimulation is split into the two components $(\mathcal{E}_1, M)$ and $(\mathcal{E}_2, N)$. When C is a context of arity $|\Gamma|$, we write $C[\Gamma]$ for the term obtained by replacing each hole $[\cdot]_i$ in C with the value Γ_i. The rules for transitions in ΛN^1 are as follows; they are reminiscent of [19].

$$\frac{M \longmapsto_{\mathsf{n}} M'}{(\Gamma, M) \xrightarrow{\tau} (\Gamma, M')} \qquad \frac{\Gamma_i(C[\Gamma]) \longmapsto_{\mathsf{n}} M'}{\Gamma \xrightarrow{i,C} (\Gamma, M')} \tag{4.1}$$

The first rule says that if M reduces to M' in ΛN then M can also reduce in ΛN^1, in any environment. The second rule implements the observations in clause (2) of Definition 17: in an environment Γ (only containing values), any component Γ_i can be tested by supplying, as input, a term obtained by filling a context C with values from Γ itself. The label of the transition records the position i and the context chosen. As the rules show, the labels of ΛN^1 include the special label τ, and can also be of the form i, C where i is a integer and C a context.

We establish full abstraction from environmental bisimilarity to bisimilarity on ΛN^1 for finite environments. Full abstraction for the empty environment alone is enough for our interests since contextual equivalence corresponds to environmental bisimilarity with the empty environment. One could accommodate ΛN^1 and the corresponding full abstraction result for possibly-infinite environments, however we felt that it was not worth the notational complications, since infinite environments are not reachable from finite ones in environmental bisimulations, and since we do not think that infinite environments increase discriminative power. In the statement below, $\approx$ denotes standard weak bisimilarity (Definition 2) on ΛN^1.

The following proof shows a precise correspondence between environmental bisimulations and bisimulations in ΛN^1. The reader familiar with environmental bisimilarities should find the statement illustrative and maybe applicable to other variants of environmental bisimilarities. It is also possible to show a direct, although less precise, correspondence between contextual equivalence and bisimilarity. This second approach is shown for the imperative λ-calculus in Section 5 and exploits the compatibility of up-to-context functions. Since compatibility of up-to-context is proved independently of the correspondence result for ΛN^1, this approach would also work for ΛN^1; however, we found it more interesting here to show the more precise result.

Theorem 18. When $\mathcal{E}$ is a finite environment,

$$M \approx_{\mathcal{E}}^{\mathsf{env}} N \;\Leftrightarrow\; (\mathcal{E}_1, M) \approx (\mathcal{E}_2, N) \qquad \text{and} \qquad \mathcal{E} \in \approx^{\mathsf{env}} \;\Leftrightarrow\; \mathcal{E}_1 \approx \mathcal{E}_2 \quad .$$

Proof. ($\Rightarrow$) We show that if $\mathcal{X}$ is an environmental bisimulation then $\mathcal{X}^2$ is a (first-order) weak bisimulation, where $\mathcal{X}^2$ relates $(\mathcal{E}_1, M)$ to $(\mathcal{E}_2, N)$ when $(\mathcal{E}, M, N) \in \mathcal{X}$, and $\mathcal{E}_1$ to $\mathcal{E}_2$ when $\mathcal{E} \in \mathcal{X}$. By symmetry we consider only one direction: we suppose $x\ \mathcal{X}^2\ y$ and a transition $x \xrightarrow{\mu} x'$, and we obtain y' such that $y \stackrel{\hat{\mu}}{\Longrightarrow} y'$ and $x'\ \mathcal{X}^2\ y'$.

(1) $\mu = \tau$: then $x = (\mathcal{E}_1, M) \xrightarrow{\tau} (\mathcal{E}_1, M') = x'$ with $M \longmapsto_{\mathsf{n}} M'$, and $y = (\mathcal{E}_2, N)$ with $M\ \mathcal{X}_{\mathcal{E}}\ N$. By definition of environmental bisimulation, $N \Longmapsto_{\mathsf{n}} N'$ with $M'\ \mathcal{X}_{\mathcal{E}}\ N'$ and hence $y \Longrightarrow y'$ with $y' = (\mathcal{E}_2, N')$ and $x'\ \mathcal{X}^2\ y'$.

(2) $\mu = i, C$: then $(x, y) = (\mathcal{E}_1, \mathcal{E}_2)$ with $\mathcal{E} \in \mathcal{X}$, and $x' = (\mathcal{E}_1, P\{C[\mathcal{E}_1]/x\})$ with $\lambda x.P = (\mathcal{E}_1)_i$ and we choose $y' = (\mathcal{E}_2, Q\{C[\mathcal{E}_2]/x\})$ with $\lambda x.Q = (\mathcal{E}_2)_i$. Then by construction, $y \xrightarrow{i,C} y'$ and $x'\ \mathcal{X}^2\ y'$ because $(\lambda x.P, \lambda x.Q) \in \mathcal{E}$ and $(C[\mathcal{E}_1], C[\mathcal{E}_2]) \in \mathcal{E}^\star$.

($\Leftarrow$) The correspondence is less direct, so instead of establishing a correspondence between weak bisimulations, we define the candidate relation on top of weak bisimilarity. We first write $\Gamma \cdot \Delta$ for the *pairing* of Γ and Δ, i.e. the relation $\{(\Gamma_i, \Delta_i) \mid i \leq |\Gamma|, |\Delta|\}$. The environmental relation $\mathcal{X}$ is defined as follows:

$$\mathcal{X} \;\triangleq\; \{(\Gamma \cdot \Delta, M, N) \mid (\Gamma, M) \approx (\Delta, N)\} \;\cup\; \{\Gamma \cdot \Delta \mid \Gamma \approx \Delta\}$$

(where Γ and Δ only contain values). We prove that $\mathcal{X}$ is an environmental bisimulation.

(1) Suppose $M\ \mathcal{X}_{\Gamma\cdot\Delta}\ N$ (i.e. $(\Gamma, M) \approx (\Delta, N)$).
   (a) if $M \longmapsto_{\mathsf{n}} M'$ then $(\Gamma, M) \xrightarrow{\tau} (\Gamma, M')$, which implies $(\Delta, N) \Longrightarrow (\Delta, N')$ with $(\Gamma, M') \approx (\Delta, N')$ and hence $N \Longmapsto_{\mathsf{n}} N'$ with $M'\ \mathcal{X}_{\Gamma\cdot\Delta}\ N'$;
   (b) if $M = V$, we need a W such that $N \Longmapsto_{\mathsf{n}} W$ and $\Gamma \cdot \Delta \cup \{(V, W)\} \in \mathcal{X}$. Since $(\Gamma, V) = x \xrightarrow{i,C} x'$ for some x', we have $(\Delta, N) \stackrel{i,C}{\Longrightarrow} y_3$, for some y_3, i.e. $(\Delta, N) = y_0 \Longrightarrow y_1 \xrightarrow{i,C} y_2 \Longrightarrow y_3$. Since y_1 has an i, C transition, y_1 is of the form (Δ, W) for some W. Since $y_0 \Longrightarrow y_1$ and $\xrightarrow{\tau}$ is deterministic, we derive $y_0 \approx y_1$. By

transitivity of $\approx$, we infer $x \approx y_1$, hence $(\Gamma, V) \approx (\Delta, W)$. We can then conclude $(\Gamma, V) \cdot (\Delta, W) \in \mathcal{X}$.

(c) the converse of the above two conditions, on N, holds, as $\approx$ is symmetric.

(2) If $(\lambda x.P, \lambda x.Q) \in \Gamma \cdot \Delta \in \mathcal{X}$ and $(M, N) \in (\Gamma \cdot \Delta)^\star$, we prove that $P\{M/x\}\ \mathcal{X}_{\Gamma\cdot\Delta}\ Q\{N/x\}$.

We have $(\lambda x.P, \lambda x.Q) = (\Gamma_i, \Delta_i)$ for some i, and $(M, N) = (C[\Gamma], C[\Delta])$ for some C. Then $\Gamma_i(C[\Gamma]) \longmapsto_{\mathsf{n}} P\{M/x\}$ so $\Gamma \xrightarrow{i,C} (\Gamma, P\{M/x\})$ must be answered with $\Delta \overset{i,C}{\Longrightarrow} (\Delta, N')$ and $(\Gamma, P\{M/x\}) \approx (\Delta, N')$. There are no silent transitions coming from Δ so we necessarily have the $\xrightarrow{i,C}$ transition first. Since there is only one such transition, we have in fact $\Delta \xrightarrow{i,C} (\Delta, Q\{N/x\}) \Longrightarrow (\Delta, N')$. Again, as $\Longrightarrow \subseteq \approx$, by transitivity of $\approx$ we derive $(\Gamma, P\{M/x\}) \approx (\Delta, N') \approx (\Delta, Q\{N/x\})$, and hence $(P\{M/x\})\ \mathcal{X}_{\Gamma\cdot\Delta}\ Q\{N/x\}$. □

The theorem also holds for the strong versions of the bisimilarities. Again, having established full abstraction with respect to a first-order transition system and ordinary bisimilarity, we can inherit the theory of bisimulation enhancements. We have however to check up-to techniques that are specific to environmental bisimilarity.

Structure and reusability of proofs. The first technique is proved compatible in Lemma 19, which is an example of the standard way of proving compatibility. The other three techniques are interdependent in that they each progress to a function containing all three (Lemmas 20, 29, and 30). These progressions could be established separately, which would be an improvement of modularity over a monolithic proof of compatibility (itself an improvement of size over two redundant proofs of up-to-context and congruence). Moreover, we achieve here a substantial amount of additional proof refactoring thanks to two general ingredients. The first (Definition 21, Lemmas 22 and 28) may be of general interest to handle calculi whose grammars separate 'values' from 'non-value'. The second (Lemmas 24, 25, and 26) may be of general interest for calculi that are quasi-deterministic, in the sense of Definition 23. (These results are used again in Section 5.) The three progressions are finally combined into Theorem 31.

A useful technique specific to environmental bisimilarity is 'up-to-environment', which allows us to replace an environment with a larger one. We define $\mathsf{w}(\mathcal{R})$ as the smallest relation that includes $\mathcal{R}$ and such that, whenever $(V, \Gamma, M)\ \mathsf{w}(\mathcal{R})\ (W, \Delta, N)$ holds, also $(\Gamma, M)\ \mathsf{w}(\mathcal{R})\ (\Delta, N)$ holds, where V and W are any values. Here w stands for 'weakening' as, from Lemmas 6 and 19, if $(V, \Gamma, M) \approx (W, \Delta, N)$ then $(\Gamma, M) \approx (\Delta, N)$.

Lemma 19. Function w is compatible.

Proof. Since silent transitions do not alter the environment, we only consider (i, C)-transitions; writing $\Gamma' = V_1, \ldots, V_n, \Gamma$ and $\Delta' = W_1, \ldots, W_n, \Delta$, we have:

$$\Gamma_i(C[\Gamma]) = \Gamma'_{i+n}(C_{+n}[\Gamma]) \qquad \Delta_i(C[\Delta]) = \Delta'_{i+n}(C_{+n}[\Delta])\ ,$$

where C_{+n} is C where each hole $[\cdot]_j$ has been replaced with $[\cdot]_{j+n}$. Then $\Gamma \xrightarrow{i,C} (\Gamma, M')$ implies $\Gamma' \xrightarrow{i+n,C_{+n}} (\Gamma', M')$ and $\Delta' \overset{i+n,C_{+n}}{\Longrightarrow} (\Delta', N')$ implies $\Delta \overset{i,C}{\Longrightarrow} (\Delta, N')$, and so from $\mathcal{R} \rightsquigarrow \mathcal{S}$ we obtain $\mathsf{w}(\mathcal{R}) \rightsquigarrow \mathsf{w}(\mathcal{S})$. □

Somewhat dual to weakening is the strengthening of the environment, in which a component of an environment can be removed. However this is only possible if the component removed is 'redundant', that is, it can be obtained by gluing other pieces of the environment within a context; strengthening is captured by the following str function:

$$\mathsf{str}(\mathcal{R}) \triangleq \big\{((\Gamma, C_v[\Gamma], M), (\Delta, C_v[\Delta], N)) \text{ s.t. } (\Gamma, M)\ \mathcal{R}\ (\Delta, N)\big\}$$

where C_v ranges over value contexts (i.e., the outermost operator of C_v is an abstraction or C_v is a hole). We show that str is below the companion in Theorem 31.

For up-to-context, we need to distinguish between arbitrary contexts and evaluation contexts. There are indeed congruence properties, and corresponding up-to techniques, that only hold for the latter contexts. A hole $[\cdot]_i$ of a context C is in a *redex position* if the context obtained by filling all the holes but $[\cdot]_i$ with values is an evaluation context. Below, C ranges over arbitrary contexts, whereas E ranges over contexts in which the first hole $[\cdot]_1$ appears exactly once *and* in redex position.

$$\begin{array}{rcll} \mathcal{C}(\mathcal{R}) & \triangleq & \big\{((\Gamma, C[\Gamma]), (\Delta, C[\Delta])) & \text{s.t. } \Gamma\ \mathcal{R}\ \Delta\ \big\} \\ \mathcal{C}_{\mathsf{e}}(\mathcal{R}) & \triangleq & \big\{((\Gamma, E[M, \Gamma]), (\Delta, E[N, \Delta])) & \text{s.t. } (\Gamma, M)\ \mathcal{R}\ (\Delta, N)\big\} \end{array}$$

We will prove that functions $\mathcal{C}$, str, and $\mathcal{C}_{\mathsf{e}}$ are below both companions with a separate progression result for each function. We start by establishing a progression for $\mathcal{C}$.

Lemma 20. $\mathcal{C} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_{\mathsf{e}})$.

Proof. Suppose that $\mathcal{R} \rightsquigarrow \mathcal{S}$, we show that $\mathcal{C}(\mathcal{R}) \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_{\mathsf{e}})(\mathcal{S})$. More explicitly, we show that $\mathcal{C}(\mathcal{R}) \rightsquigarrow \mathsf{str}(\mathcal{S}) \cup \mathsf{str}(\mathcal{C}(\mathcal{R})) \cup \mathcal{C}(\mathcal{R}) \cup \mathcal{C}_{\mathsf{e}}(\mathcal{S})$.

Let $\Gamma\ \mathcal{R}\ \Delta$ and $n = |\Gamma|$. We analyse transitions from $(\Gamma, C[\Gamma])$.

(1) Suppose $C[\Gamma]$ is a value, so C is a value context C_v; the transition to consider is of the form $(\Gamma, C_v[\Gamma]) \xrightarrow{i,C} (\Gamma, C_v[\Gamma], M')$.
   (a) If $i \leq n$ then $\Gamma_i(C[\Gamma, C_v[\Gamma]]) \longmapsto_{\mathsf{n}} M'$ and since $C[\Gamma, C_v[\Gamma]] = C'[\Gamma]$ with $C' = C[-, C_v]$, we obtain $\Gamma \xrightarrow{i,C'} (\Gamma, M')$, and similarly for Δ.

$$\begin{array}{ccc} \Gamma & \mathcal{R} & \Delta \\ {\scriptstyle i,C'}\downarrow & & \Downarrow{\scriptstyle i,C'} \\ (\Gamma, M') & \mathcal{S} & (\Delta, N') \end{array} \quad \rightsquigarrow \quad \begin{array}{ccc} (\Gamma, C_v[\Gamma]) & \mathcal{C}(\mathcal{R}) & (\Delta, C_v[\Delta]) \\ {\scriptstyle i,C}\downarrow & & \Downarrow{\scriptstyle i,C} \\ (\Gamma, C_v[\Gamma], M') & \mathsf{str}(\mathcal{S}) & (\Delta, C_v[\Delta], N') \end{array}$$

   (b) If $i = n + 1$ and $C_v = [\cdot]_j$, the same argument as above applies, replacing Γ_i with Γ_j and i, C' with j, C'.
   (c) If $i = n + 1$ and C_v is not a hole, then M' is of the form $C'[\Gamma]$; then Δ makes the same transition to $(\Delta, C_v[\Delta], C'[\Delta])$ and $((\Gamma, C_v[\Gamma], C'[\Gamma]), (\Delta, C_v[\Delta], C'[\Delta])) \in \mathsf{str}(\mathcal{C}(\mathcal{R}))$.

(2) If $C[\Gamma]$ is not a value then C is necessarily of the form $C = E[C_{v1}C_2, -]$, for some evaluation context E, value context C_{v1}, and context C_2. The transition is of the form $(\Gamma, C[\Gamma]) \xrightarrow{\tau} (\Gamma, M')$ with $C[\Gamma] \longmapsto_{\mathsf{n}} M'$. We distinguish two cases:
   (a) C_{v1} is not a hole (i.e. $C_{v1} = \lambda x.C_1[x, -]$ for some C_1). Then $M' = C'[\Gamma]$ for $C' = E[C_1[C_2, -], -]$ and $C[\Delta] \longmapsto_{\mathsf{n}} C'[\Delta]$. The resulting pair $((\Gamma, C'[\Gamma]), (\Delta, C'[\Delta]))$ is in $\mathcal{C}(\mathcal{R})$.
   (b) $C_{v1} = [\cdot]_i$ and so $C[\Gamma] = E[\Gamma_i(C_2[\Gamma]), \Gamma]$. Then $M' = E[M_1, \Gamma]$ for some M_1 such that $\Gamma_i(C_2[\Gamma]) \longmapsto_{\mathsf{n}} M_1$. Using the label i, C_2, the progression $\mathcal{R} \rightsquigarrow \mathcal{S}$ provides us

with an answer (Δ, N_1) such that $\Delta_i(C_2[\Delta]) \Longrightarrow_{\mathsf{n}} N_1$, which allows us to conclude up to $\mathcal{C}_{\mathsf{e}}$:

$$\begin{array}{ccc} \Gamma & \mathcal{R} & \Delta \\ {\scriptstyle i,C_2}\downarrow & & \Downarrow{\scriptstyle i,C_2} \\ (\Gamma, M_1) & \mathcal{S} & (\Delta, N_1) \end{array} \quad \rightsquigarrow \quad \begin{array}{ccc} (\Gamma, E[\Gamma_i(C_2[\Gamma]), \Gamma]) & \mathcal{C}(\mathcal{R}) & (\Delta, E[\Delta_i(C_2[\Delta]), \Delta]) \\ {\scriptstyle \tau}\downarrow & & \Downarrow{\scriptstyle \tau} \\ (\Gamma, E[M_1, \Gamma]) & \mathcal{C}_{\mathsf{e}}(\mathcal{S}) & (\Delta, E[N_1, \Delta]) \end{array}$$

□

Before moving on to the techniques str and $\mathcal{C}_{\mathsf{e}}$, it is useful to remark that when they are applied to values, they look like special cases of $\mathcal{C}$. This can be used to shorten the proofs substantially, but this needs to be made formal first by defining a restriction function and using it to relate str and $\mathcal{C}_{\mathsf{e}}$ to $\mathcal{C}$.

Definition 21. Let $\mathcal{V}$ be the set of value configurations (of form Γ) and $\overline{\mathcal{V}}$ the set of non-value configurations, i.e. sequences for which the last term is not a value (of form (Γ, M) where M is not a value). We define now two restriction functions on relations:

$$\mathsf{v}(\mathcal{R}) \triangleq \mathcal{R} \cap (\mathcal{V} \times \mathcal{V})$$
$$\mathsf{n}(\mathcal{R}) \triangleq \mathcal{R} \cap (\overline{\mathcal{V}} \times \overline{\mathcal{V}})$$

The first step is to show that indeed, techniques $\mathcal{C}_{\mathsf{e}}$ and str are, on value configuration pairs, special cases of $\mathcal{C}$:

Lemma 22. $\mathcal{C}_{\mathsf{e}} \circ \mathsf{v} \subseteq t \circ \mathcal{C}$ and $\mathsf{str} \circ \mathsf{v} \subseteq t \circ \mathcal{C}$.

Proof. Any pair in $\mathcal{C}_{\mathsf{e}}(\mathsf{v}(\mathcal{R}))$ is of the form $((\Gamma', E[\Gamma_n, \Gamma']), (\Delta', E[\Delta_n, \Delta']))$ where: n is the arity of E, $(\Gamma, \Delta) \in \mathcal{R}$, and Γ' (respectively Δ') is the sequence Γ (respectively Δ) without its last element. The context $C = E[[\cdot]_n, [\cdot]_1, \ldots, [\cdot]_{n-1}]$ applied to $(\Gamma, \Delta) \in \mathcal{R}$ shows that the original pair is of the form $((\Gamma', C[\Gamma]), (\Delta', C[\Delta]))$ and hence is in $t(\mathcal{C}(\mathcal{R}))$: we use $\mathsf{w} \subseteq t$ to remove the nth values from the environments Γ and Δ. The same argument applies for str as well, except that we use t in $t(\mathcal{C}(\mathcal{R}))$ only to swap the last two elements the sequences. □

We handled pairs of value configurations, so now we need to handle the other kinds of pairs. We first handle the case where the left member of the pair is a value configuration. We need however to first define a notion of determinism of an LTS:

Definition 23. We say that a LTS $(Pr, Act, \longrightarrow)$ is *quasi-deterministic* if there exists an equivalence relation $\simeq$ on Act such that for all labels $\mu, \mu' \in Act$ and processes $x, x_1, x_2 \in Pr$ (where $x \xrightarrow{\mu}$ is short for $(\exists x'\ x \xrightarrow{\mu} x')$):

(1) $\mu \simeq \tau$ implies $\mu = \tau$,
(2) $x \xrightarrow{\mu} x_1$ and $x \xrightarrow{\mu} x_2$ imply $x_1 \sim x_2$,
(3) $x \xrightarrow{\mu}$ and $x \xrightarrow{\mu'}$ imply $\mu \simeq \mu'$,
(4) $x \xrightarrow{\mu}$ and $\mu \simeq \mu'$ implies $x \xrightarrow{\mu'}$.

This version of determinism is looser than strict determinism, since it allows derivatives to be strongly bisimilar and not necessarily equal, and labels to be related through some

equivalence relation, rather than equal. This equivalence relation must in turn be reflected by the set of labels that can be performed from a given process.

A similar notion can be found in the formalisation of a compiler with some non-determinism [38], where such a relation on labels is defined. This relation satisfies (1), a LTS that is said to be 'determinate' satisfies (2) (although (2) is more relaxed as it allows for bisimilar processes) and (3) and a 'receptive' LTS satisfies (4).

Lemma 24. In a quasi-deterministic LTS, $\xrightarrow{\tau} \subseteq \gtrsim$.

Proof. We show that $(\xrightarrow{\tau}) \subseteq \mathbf{ep}(\gtrsim)$. Let x, y such that $x \xrightarrow{\tau} y$.

- If $x \xrightarrow{\mu} x'$, then $\mu \simeq \tau$ by (3), $\mu = \tau$ by (1), $x' \sim y$ by (2), so in particular $x' \gtrsim y$. Hence, the challenge can be answered with $y \stackrel{\hat{\mu}}{\Longrightarrow} y$.
- If $y \xrightarrow{\mu} y'$, then $x \stackrel{\mu}{\Longrightarrow} y'$, so we conclude by reflexivity of $\gtrsim$.

We conclude by remarking that $\mathbf{ep}(\gtrsim) \subseteq (\gtrsim)$. □

Lemma 25. In a quasi-deterministic LTS, if $(x, y) \in \mathbf{wp}(\mathcal{S})$ and $x \xrightarrow{\mu}$ with $\mu \neq \tau$, then for some y_1, $y \Longrightarrow y_1 \xrightarrow{\mu}$ with $(x, y_1) \in \mathbf{wp}(\gtrsim \mathcal{S} \lesssim)$.

Proof. We first prove that whenever $\mu_1, \mu_2 \neq \tau$, for all x_2, x_2',

$$(x_1 \Longrightarrow x_1' \xrightarrow{\mu_1}) \wedge (x_2 \Longrightarrow x_2' \xrightarrow{\mu_2}) \wedge x_1 \sim x_2 \Rightarrow x_1' \sim x_2' \tag{4.2}$$

by induction on $x_1 \Longrightarrow x_1'$.

- If $x_1 = x_1'$, it is enough to show that $x_2 = x_2'$. Suppose otherwise that $x_2 \Longrightarrow x_2'$ takes at least one step, and so $x_2 \xrightarrow{\tau}$. Since $x_1 \sim x_2$, $x_1 \xrightarrow{\tau}$, and so by (3), $\mu_1 \simeq \tau$, and by (1), $\mu_1 = \tau$ (contradiction).
- Suppose now $x_1 \xrightarrow{\tau} x_1' \Longrightarrow x_1''$, and that the induction hypothesis holds for x_1'. Since $x_1 \sim x_2$, we can derive x_2' such that $x_2 \xrightarrow{\tau} x_2'$ and $x_1' \sim x_2'$. The transition $x_2 \Longrightarrow x_2''$ must take at least one step to some x', otherwise by (3), $\mu_2 \simeq \tau$ and then by (1), $\mu_2 = \tau$ (contradiction). By (2), $x' \sim x_2'$, and by transitivity and symmetry of bisimilarity, $x_1' \sim x'$, so we conclude by induction.

We have established (4.2).

Since $x \xrightarrow{\mu}$, $(x, y) \in \mathbf{wp}(S)$ provides us with y_1, y_1', y_1'' such that $y \Longrightarrow y_1 \xrightarrow{\hat{\mu}} y_1' \Longrightarrow y_1''$. Because $\mu \neq \tau$, $y_1 \xrightarrow{\mu} y_1'$, so there only remains to prove that $(x, y_1) \in \mathbf{wp}(\gtrsim \mathcal{S} \lesssim)$. More precisely we will prove that $(x, y_1) \in \mathbf{wp}(\mathcal{S} \sim \Leftarrow)$, which entails $(x, y_1) \in \mathbf{wp}(\gtrsim \mathcal{S} \lesssim)$ by Lemma 24. Note that by (4.2), for all $\alpha \neq \tau$ and y_0, $y \Longrightarrow y_0 \xrightarrow{\alpha}$ implies $y_1 \sim y_0$ $(*)$. We now prove (4.3), which we will use twice.

$$x \xrightarrow{\alpha} x_2 \;\Rightarrow\; \exists y_2''\; y_1'\; y_1''\; y_1 \xrightarrow{\alpha} y_1' \Longrightarrow y_1'' \sim y_2'' \;\wedge\; x_2 \,\mathcal{S}\, y_2'' \tag{4.3}$$

By (3) and (1), $\alpha \neq \tau$. $\mathbf{wp}(\mathcal{S})$ provides us again with y_2, y_2', y_2'' such that $y \Longrightarrow y_2 \xrightarrow{\alpha} y_2' \Longrightarrow y_2''$ with $x_2 \,\mathcal{S}\, y_2''$. By $(*)$, $y_1 \sim y_2$, from which we can play the transitions $y_2 \xrightarrow{\alpha} \Longrightarrow y_2''$ to obtain y_1'' such that $y_1 \xrightarrow{\alpha} \Longrightarrow y_1''$ with $y_1'' \sim y_2''$, which ends the proof of (4.3).

We finally show $(x, y_1) \in \mathbf{wp}(\mathcal{S} \sim \Leftarrow)$:

- Suppose that $x \xrightarrow{\alpha} x_2$. Using (4.3), we can answer the challenge with y_1''. We indeed have $y_1 \stackrel{\alpha}{\Longrightarrow} y_1''$ and $x_2 \,\mathcal{S} \sim y_1''$, hence $x_2 \,\mathcal{S} \sim \Leftarrow y_1''$.

- Suppose now that $y_1 \xrightarrow{\alpha} y_3'$. By (3), $\mu \simeq \alpha$, and by (4), $x \xrightarrow{\alpha} x_2$ for some x_2. We use $x_2 \xrightarrow{\alpha} x_2'$ as our weak transition. We can now use (4.3) again. By (2), $y_1' \sim y_3'$. Since $y_1' \Longrightarrow y_1''$, there is y_3'' such that $y_3' \Longrightarrow y_3''$ and $y_1'' \sim y_3''$. We conclude by transitivity of $\sim$ since $y_2' \sim y_1'' \sim y_3''$.

□

Lemma 26. In ΛN^1, if $(\Gamma, y) \in \mathbf{wp}(\mathcal{S})$ then $y \Longrightarrow \Delta$ with $(\Gamma, \Delta) \in \mathbf{wp}(\gtrsim \mathcal{S} \lesssim)$.

Proof. ΛN^1 is quasi-deterministic, using $i, C \simeq i', C'$ whenever C and C' are of the same arity, so we use Lemma 25 with $x = \Delta$, $\mu = 1, C_0$ (where C_0 is a context of arity $|\Gamma|$, for example the context $\lambda x.x$ with no hole), which gives us $y \Longrightarrow y' \xrightarrow{\mu}$, hence y' is of the form Δ, with $(x, y') \in \mathbf{wp}(\gtrsim \mathcal{S} \lesssim)$. □

Remark 27. Lemma 26 does not apply to non-deterministic calculi. In fact, those would require a special label signalling that the configuration is only composed of values in order for the proofs of progressions to go through. This would make Lemma 26 unnecessary in the proofs of progressions for str and $\mathcal{C}_\mathsf{e}$ (those proofs would however need to include long parts that are redundant with the proof of progression for $\mathcal{C}$ since Lemma 28 uses Lemma 26).

Lemma 28 helps separating a proof of progression $f \rightsquigarrow T(f \cup g)$ for a function f into a few simpler proofs, namely: (4.4), that is, the progression for pairs of values; (4.5), that is, the progression for pairs of non-values; and (4.6), that is, the fact that f absorbs the reduction function $\mathsf{r} \triangleq (\mathcal{R} \mapsto \Rightarrow\mathcal{R}\Leftarrow)$, up to t. In order to carry out the splitting, f is also required to distribute over union (4.7) — which holds for str, $\mathcal{C}$, and $\mathcal{C}_\mathsf{e}$.

Lemma 28. If f and g are monotone functions such that:

$$f \circ \mathsf{v} \rightsquigarrow T(f \cup g) \tag{4.4}$$

$$f \circ \mathsf{n} \rightsquigarrow T(f \cup g) \tag{4.5}$$

$$f \circ \mathsf{r} \subseteq t \circ f \tag{4.6}$$

$$f(\mathcal{R} \cup \mathcal{S}) \subseteq f(\mathcal{R}) \cup f(\mathcal{S}) \tag{4.7}$$

then $f \rightsquigarrow T(f \cup g)$.

Proof. We first establish the following inclusion:

$$(\mathsf{id} \setminus \mathsf{n}) \circ \mathbf{p} \subseteq \mathsf{r} \circ \mathsf{v} \circ \mathbf{p} \circ t \tag{4.8}$$

Let $\mathcal{S}$ be a relation and $(x, y) \in (\mathsf{id} \setminus \mathsf{n})(\mathbf{p}(\mathcal{S}))$, i.e. $(x, y) \in \mathbf{p}(\mathcal{S})$ and at least one of x or y is a value. The case $\mathbf{p} = \mathbf{sp}$ is trivial: since x is a value if and only if y is a value, they are both values, and $(x, y) \in \mathsf{v}(\mathbf{p}(\mathcal{S})) \subseteq \mathsf{r}(\mathsf{v}(\mathbf{p}(t(\mathcal{S}))))$. The case $\mathbf{p} = \mathbf{wp}$ is a consequence of Lemma 26:

- First suppose that x is a value Γ. By Lemma 26, there is a value Δ such that $(\Gamma, \Delta) \in \mathbf{wp}(\gtrsim \mathcal{S} \lesssim) \subseteq \mathbf{wp}(t(\mathcal{S}))$. This is a value pair, so $(\Gamma, \Delta) \in \mathsf{v}(\mathbf{wp}(t(\mathcal{S})))$. Finally, $(x, y) \in \mathsf{r}(\mathsf{v}(\mathbf{wp}(t(\mathcal{S}))))$.
- Otherwise, suppose y is a value Δ. We know that $(\Delta, y) \in \mathbf{wp}(\mathcal{S})^{-1} = \mathbf{wp}(\mathcal{S}^{-1})$ by symmetry of $\mathbf{wp}$, so we can apply Lemma 26. This shows that there exists Γ such that $(\Delta, \Gamma) \in \mathbf{wp}(\gtrsim \mathcal{S}^{-1} \lesssim)$. Following the reasoning for x, we derive $(y, x) \in \mathsf{r}(\mathsf{v}(\mathbf{wp}(t(\mathcal{S}^{-1}))))$, and so $(x, y) \in \mathsf{r}(\mathsf{v}(\mathbf{wp}(t(\mathcal{S}))))$ since r, v, $\mathbf{wp}$, and t are symmetric.

We can now conclude:

$$\begin{aligned}
f \circ \mathbf{p} &= f \circ (\mathsf{n} \cup (\mathsf{id} \setminus \mathsf{n})) \circ \mathbf{p} \\
&\subseteq f \circ \mathsf{n} \circ \mathbf{p} \cup f \circ (\mathsf{id} \setminus \mathsf{n}) \circ \mathbf{p} && \text{by (4.7)} \\
&\subseteq \mathbf{p} \circ T(f \cup g) \cup f \circ (\mathsf{id} \setminus \mathsf{n}) \circ \mathbf{p} && \text{by (4.5)} \\
&\subseteq \mathbf{p} \circ T(f \cup g) \cup f \circ \mathsf{r} \circ \mathsf{v} \circ \mathbf{p} \circ t && \text{by (4.8) and monotonicity of } f \\
&\subseteq \mathbf{p} \circ T(f \cup g) \cup t \circ f \circ \mathsf{v} \circ \mathbf{p} \circ t && \text{by (4.6)} \\
&\subseteq \mathbf{p} \circ T(f \cup g) \cup t \circ \mathbf{p} \circ T(f \cup g) \circ t && \text{by (4.4) and monotonicity of } t \\
&\subseteq \mathbf{p} \circ T(f \cup g) \cup \mathbf{p} \circ t \circ T(f \cup g) \circ t && \text{by compatibility of } t \\
&= \mathbf{p} \circ T(f \cup g) && \text{since } t \subseteq T(h) \text{ and } T(h)^3 = T(h) \text{ for all } h
\end{aligned}$$

□

The distinction between values and non-values simplifies the proof of the progression for str.

Lemma 29. $\mathsf{str} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e})$.

Proof. This is the conclusion of Lemma 28 with $f = \mathsf{str}$ and $g = \mathcal{C} \cup \mathcal{C}_\mathsf{e}$, so it is sufficient to establish the premises of the lemma:

(1) $\mathsf{str} \circ \mathsf{v} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e})$:

$$\begin{aligned}
\mathsf{str} \circ \mathsf{v} \circ \mathbf{p} &\subseteq t \circ \mathcal{C} \circ \mathbf{p} && \text{by Lemma 22} \\
&\subseteq t \circ \mathbf{p} \circ T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e}) && \text{by Lemma 20 and monotonicity of } t \\
&\subseteq \mathbf{p} \circ t \circ T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e}) && \text{by compatibility of } t \\
&\subseteq \mathbf{p} \circ T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e}) && \text{since } t \subseteq T(h) \text{ and } T(h)^2 = T(h) \text{ for all } h.
\end{aligned}$$

(2) $\mathsf{str} \circ \mathsf{n} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e})$ follows from the stronger inclusion $\mathsf{str} \circ \mathsf{n} \rightsquigarrow \mathsf{str}$:
Let $\mathcal{R} \rightsquigarrow \mathcal{S}$ and $((\Gamma, C_v[\Gamma], M), (\Delta, C_v[\Delta], N)) \in \mathsf{str}(\mathsf{n}(\mathcal{R}))$ i.e. with M and N non-values. Challenges from the left-hand side are of the form $(\Gamma, C_v[\Gamma], M) \xrightarrow{\tau} (\Gamma, C_v[\Gamma], M')$, which is equivalent to $(\Gamma, M) \xrightarrow{\tau} (\Gamma, M')$. The progression $\mathcal{R} \rightsquigarrow \mathcal{S}$ tells us that there exists N' such that $(\Delta, N) \Longrightarrow (\Delta, N')$ with $(\Gamma, M)\ \mathcal{S}\ (\Delta, N')$, and so $(\Delta, C_v[\Delta], N) \Longrightarrow (\Delta, C_v[\Delta], N')$ with $((\Gamma, C_v[\Gamma], M'), (\Delta, C_v[\Delta], N')) \in \mathsf{str}(\mathcal{S})$. Challenge from the right-hand side are handled symmetrically.

(3) $\mathsf{str} \circ \mathsf{r} \subseteq t \circ \mathsf{str}$: since $\mathsf{r} \subseteq f_{\gtrsim} \subseteq t$, we only need to prove $\mathsf{str} \circ \mathsf{r} \subseteq \mathsf{r} \circ \mathsf{str}$. This can be derived more algebraically: it is trivial to check that str respects relation mirroring, relational composition, and silent transitions ($\mathsf{str}(\mathcal{R}^{-1}) \subseteq \mathsf{str}(\mathcal{R})^{-1}$, $\mathsf{str}(\mathcal{R}\mathcal{S}) \subseteq \mathsf{str}(\mathcal{R})\mathsf{str}(\mathcal{S})$, and $\mathsf{str}(\xrightarrow{\tau}) \subseteq \xrightarrow{\tau}$), from which $\mathsf{str}(\Rightarrow\mathcal{R}\Leftarrow) \subseteq \Rightarrow\mathsf{str}(\mathcal{R})\Leftarrow$ is a direct consequence.

□

The stronger result $\mathsf{str} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C})$ holds as well, but it requires a longer proof (also redundant with the progression for $\mathcal{C}$) and it is not necessary. The progression for $\mathcal{C}_\mathsf{e}$ follows the same pattern.

Lemma 30. $\mathcal{C}_\mathsf{e} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e})$.

Proof. Similarly we apply Lemma 28 with $f = \mathcal{C}_\mathsf{e}$ and $g = \mathcal{C} \cup \mathsf{str}$, and prove the hypotheses:

(1) $\mathcal{C}_\mathsf{e} \circ \mathsf{v} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e})$ is a consequence of Lemmas 22 and 20.

(2) $\mathcal{C}_\mathsf{e} \circ \mathsf{n} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_\mathsf{e})$ follows from the stronger inclusion $\mathcal{C}_\mathsf{e} \circ \mathsf{n} \rightsquigarrow \mathcal{C}_\mathsf{e}$. This can be proved the same way as in Lemma 29 using the facts that $E[M, \Gamma] \longmapsto_\mathsf{n} M_1$ implies that for some M', $M_1 = E[M', \Gamma]$ with $M \longmapsto_\mathsf{n} M'$ and that $N \Longmapsto_\mathsf{n} N'$ implies $E[N, \Delta] \Longmapsto_\mathsf{n} E[N', \Delta]$.

(3) $\mathcal{C}_{\mathsf{e}} \circ \mathsf{r} \subseteq t \circ \mathcal{C}_{\mathsf{e}}$ is similarly derived from $\mathcal{C}_{\mathsf{e}}(\mathcal{R}^{-1}) \subseteq \mathcal{C}_{\mathsf{e}}(\mathcal{R})^{-1}$, $\mathcal{C}_{\mathsf{e}}(\mathcal{RS}) \subseteq \mathcal{C}_{\mathsf{e}}(\mathcal{R})\mathcal{C}_{\mathsf{e}}(\mathcal{S})$, and $\mathcal{C}_{\mathsf{e}}(\xrightarrow{\tau}) \subseteq \xrightarrow{\tau}$.

□

Theorem 31. The functions $\mathsf{str}, \mathcal{C}, \mathcal{C}_{\mathsf{e}}$ are below both companions $t_{\mathbf{sp}}$ and $t_{\mathbf{wp}}$.

Proof. Combining Lemmas 20, 29, and 30 provides us with the following progression for **wp**:

$$\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_{\mathsf{e}} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_{\mathsf{e}})$$

and therefore $\mathsf{str} \cup \mathcal{C} \cup \mathcal{C}_{\mathsf{e}}$ is below the companion $t_{\mathbf{wp}}$. The case for **sp** is similar but easier; in particular the analogue of Lemma 26 is not required. □

Once more, the fact that up-to-context functions are below t entails the corresponding congruence properties of environmental bisimilarity. In [34] the two aspects (congruence and up-to-context) had to be proved separately, with similar proofs. Moreover the two cases of contexts (arbitrary contexts and evaluation contexts) had to be considered at the same time, within the same proof. Here, in contrast, the machinery of compatible functions allows us to split the effort into simpler proofs.

Remark 32. A transition system ensuring full abstraction as in Theorem 18 does not guarantee the compatibility of the up-to techniques specific to the language in consideration. For instance, a simpler and maybe more natural alternative to the second transition in (4.1) is the following one:

$$\frac{}{\Gamma \xrightarrow{i,C} (\Gamma, \Gamma_i(C[\Gamma]))} \tag{4.9}$$

With this rule, full abstraction holds, but up-to-context is unsound: for every Γ and Δ, the singleton relation $\{(\Gamma, \Delta)\}$ is a bisimulation up to $\mathcal{C}$: indeed, using rule (4.9), the derivatives of the pair Γ, Δ are of the shape $\Gamma_i(C[\Gamma])$, $\Delta_i(C[\Delta])$, and they can be discarded immediately, up to the context $[\cdot]_i C$. If up-to-context were sound then we would deduce that any two terms are bisimilar. (The rule in (4.1) prevents such a behaviour since it ensures that the tested values are 'consumed' immediately.)

5. Imperative call-by-value λ-calculus

In this section we study the addition of imperative features (higher-order references, that we call locations), to a call-by-value λ-calculus. It is known that finding powerful reasoning techniques for imperative higher-order languages is a hard problem. The language, ΛR, is a simplified variant of that in [15, 34]. The syntax of terms, values, and evaluation contexts, as well as the reduction semantics are given in Figure 2. A λ-term M is run in a *store*: a partial function from locations to closed values, whose domain includes all free locations of both M and its own co-domain. We use letters r, s, u, v to range over stores. New store locations may be created using the operator $\nu\ell\, M$; the content of a store location ℓ may be read using $\mathsf{get}_\ell V$, or rewritten using $\mathsf{set}_\ell V$ (the argument of the former instruction is ignored, and the latter instruction returns the identity value $I \triangleq \lambda x.x$). We denote the reflexive and transitive closure of $\longmapsto_{\mathsf{R}}$ by $\Longmapsto_{\mathsf{R}}$.

Note that in contrast with the languages in [15, 34], locations are not directly first-class values; the expressive power is however the same: a first-class location ℓ can always be encoded as the pair $(\mathsf{get}_\ell, \mathsf{set}_\ell)$. Having locations as first-class values by themselves is

$$M ::= x \mid MM \mid \nu\ell\, M \mid V \qquad V ::= \lambda x.M \mid \mathsf{get}_\ell \mid \mathsf{set}_\ell \qquad E ::= [\cdot] \mid EV \mid ME$$

$$\frac{}{(s; (\lambda x.M)V) \longmapsto_{\mathsf{R}} (s; M\{V/x\})} \qquad \frac{\ell \notin \mathsf{dom}(s)}{(s; \nu\ell\, M) \longmapsto_{\mathsf{R}} (s[\ell \mapsto I]; M)}$$

$$\frac{\ell \in \mathsf{dom}(s)}{(s; \mathsf{get}_\ell V) \longmapsto_{\mathsf{R}} (s; s[\ell])} \qquad \frac{\ell \in \mathsf{dom}(s)}{(s; \mathsf{set}_\ell V) \longmapsto_{\mathsf{R}} (s[\ell \mapsto V]; I)} \qquad \frac{(s; M) \longmapsto_{\mathsf{R}} (s'; M')}{(s; E[M]) \longmapsto_{\mathsf{R}} (s'; E[M'])}$$

Figure 2: The imperative λ-calculus

possible but would require two additional labels (for reading and writing), two additional rules, two new cases in the corresponding case analyses, and new ways to build contexts from environments; presentation and proofs would then be substantially more involved. Hence, for readability issues, we have preferred to forbid it.

We present the first-order LTS for ΛR, and then we relate the resulting strong and weak bisimilarities directly with contextual equivalence (the reference equivalence in λ-calculi). Alternatively, we could have related the first-order bisimilarities to the environmental bisimilarities of ΛR, and then inferred the correspondence with contextual equivalence from known results about environmental bisimilarity, as we did for ΛN.

We write $(s; M)\downarrow$ when M is a value; and $(s; M)\Downarrow$ if $(s; M) \Longrightarrow_{\mathsf{R}}\downarrow$. For the definition of contextual equivalence, we distinguish the cases of values and of arbitrary terms, because they have different congruence properties: values can be tested in arbitrary contexts, while arbitrary terms must be tested only in evaluation contexts. As in [34], we consider contexts that do not contain free locations (they can contain bound locations). We refer to [34] for more details on these aspects.

Definition 33.
- For values V, W, we write $(s; V) \equiv (r; W)$ when $(s; C[V])\Downarrow$ iff $(r; C[W])\Downarrow$, for all location-free contexts C.
- For terms M and N, we write $(s; M) \equiv (r; N)$ when $(s; E[M])\Downarrow$ iff $(r; E[N])\Downarrow$, for all location-free evaluation contexts E.

We now define ΛR^1, the first-order LTS for ΛR. The states and the transitions for ΛR^1 are similar to those for the pure λ-calculus of Section 4, with the addition of a component for the store. The two transitions (4.1) of call-by-name λ-calculus become:

$$\frac{(s; M) \longmapsto_{\mathsf{R}} (s'; M')}{(s; \Gamma, M) \xrightarrow{\tau} (s'; \Gamma, M')} \qquad \frac{\Gamma' = \Gamma, \mathsf{getset}(r) \quad \big(s \uplus r[\Gamma']; \Gamma_i(C[\Gamma'])\big) \longmapsto_{\mathsf{R}} (s'; M')}{(s; \Gamma) \xrightarrow{i, C, \mathsf{cod}(r)} (s'; \Gamma', M')}$$

The first rule is the analogous of the first rule in (4.1). The important differences are on the second rule. First, since we are *call-by-value*, C now ranges over $\mathbb{C}_v$, the set of *value contexts* (i.e., holes or contexts of the form $\lambda x.C'$) without free locations. Moreover, since we are now *imperative*, in a transition we must permit the creation of new locations, and a term supplied by the environment should be allowed to use them. In the rule, the new store is represented by r (whose domain has to be disjoint from that of s). Correspondingly, to allow manipulation of these locations from the observer, for each new location ℓ we make get_ℓ and set_ℓ available, as an extension of the environment; in the rule, these are collectively written $\mathsf{getset}(r)$, and Γ' is the extended environment. Finally, we must initialise the new store, using terms that are created out of the extended environment Γ'; that is, each new

location ℓ is initialised with a term $D_\ell[\Gamma']$ (for $D_\ell \in \mathbb{C}_v$). Moreover, the contexts D_ℓ chosen must be made visible in the label of the transition. To take care of these aspects, we view r as a *store context*, a tuple of assignments $\ell \mapsto D_\ell$. Thus the initialisation of the new locations is written $r[\Gamma']$; and, denoting by $\mathtt{cod}(r)$ the tuple of the contexts D_ℓ in r, we add $\mathtt{cod}(r)$ to the label of the transition. Note also that, although C and D_ℓ are location-free, their holes may be instantiated with terms involving the get_ℓ and set_ℓ operators, so that these contexts may still manipulate the store.

Once more, on the (strong and weak) bisimilarities that are derived from this first-order LTS, we can import the theory of compatible functions and bisimulation enhancements. Like in Section 3 for π, we establish the validity of a few up-to techniques before proving full abstraction: these techniques give us important closure properties of bisimilarities via Lemma 6.

Concerning additional up-to functions, specific to ΛR^1, the functions w, str, $\mathcal{C}$ and $\mathcal{C}_\mathsf{e}$ are adapted from Section 4 in the expected manner—contexts C_v, C and E must be location-free. A further function for ΛR^1 is store, which manipulates the store by removing locations that do not appear elsewhere (akin to garbage collection); thus, $\mathsf{store}(\mathcal{R})$ is the set of all pairs

$$((s \uplus r[\Gamma'];\Gamma', M),\ (u \uplus r[\Delta'];\Delta', N))$$

such that $(s;\Gamma,M)\ \mathcal{R}\ (u;\Delta,N)$, and with $\Gamma' = \Gamma, \mathsf{getset}(r)$ and $\Delta' = \Delta, \mathsf{getset}(r)$. Note that we must have $\mathsf{dom}(r) \cap \mathsf{dom}(s) = \emptyset = \mathsf{dom}(r) \cap \mathsf{dom}(u)$. This may seem unnecessarily restrictive, but since renaming locations on either side using an injective substitution is a strongly bisimilar operation, using $(\mathcal{R} \mapsto {\sim}\mathcal{R}{\sim}) \circ \mathsf{store}$ allows to choose r_1 on the left and r_2 on the right, as long as $\mathtt{cod}(r_1) = \mathtt{cod}(r_2)$.

Lemma 34. The functions $\mathsf{w}, \mathsf{str}, \mathcal{C}_\mathsf{e}, \mathsf{store}, \mathcal{C}$ are below both companions $t_{\mathbf{sp}}$ and $t_{\mathbf{wp}}$.

Proof. We apply the same proof schema as in Theorem 31 with more technical details to be handled, as the store is to be accounted for. We provide details mainly for the progression starting from store itself, which is the most interesting new aspect. We explain how the other parts are handled, with reference to the proof of Theorem 31.

We handle store first. To avoid introducing and remembering many new names such as Γ', Γ'', etc., we write Γ^V for Γ, V and Γ^r for $\Gamma, \mathsf{getset}(r)$. For example, the rule for visible transitions can be rewritten into

$$\frac{(s;\Gamma_i(C[\Gamma^r])) \longmapsto_\mathsf{R} (s';M')}{(s,\Gamma) \xrightarrow{i,C,\mathtt{cod}(r)} (s' \uplus r[\Gamma^r],\Gamma^r,M')}\ .$$

It also simplifies writing and reading when taking one index of a composed environment, for example Γ_i^{rV} should be read as the ith element of $(\Gamma^r)^V$, which can be either Γ_i (if $i \le |\Gamma|$), or in $\mathsf{getset}(r)$, or V. Now store can be redefined as

$$\frac{(s;\Gamma,M)\ \mathcal{R}\ (u;\Delta,N)}{(s \uplus r[\Gamma^r];\Gamma^r,M)\ \mathsf{store}(\mathcal{R})\ (u \uplus r[\Delta^r];\Delta^r,N)}\ .$$

We assume $\mathcal{R} \rightsquigarrow \mathcal{S}$ and analyse the transitions starting from pairs in $\mathsf{store}(\mathcal{R})$, i.e. the transitions of $(s \uplus r[\Gamma^r];\Gamma^r,M)$. Silent transitions are, once again, easy to handle, as the locations of M are contained in the domain of s, and the other part of the term, namely $r[\Gamma^r]$, is left unchanged (progressing to $\mathsf{store}(\mathcal{S})$—in particular, $\mathsf{store} \circ \mathsf{n} \rightsquigarrow \mathsf{store}$).

We now handle the visible transitions of $(s \uplus r[\Gamma^r]; \Gamma^r, M)$, (i.e. M is a value V), labelled by μ such that $\mu = i, C, \mathtt{cod}(v)$ for some v. We choose v such that the locations used by v, which do not appear in the label, are fresh. The transition is:

$$\frac{(s \uplus r[\Gamma^r] \uplus v[\Gamma^{rVv}]; \Gamma_i^{rV}(C[\Gamma^{rVv}])) \longmapsto_{\mathsf{R}} (s'; M')}{(s \uplus r[\Gamma^r]; \Gamma^{rV}) \xrightarrow{i,C,\mathtt{cod}(v)} (s'; \Gamma^{rVv}, M')} \tag{5.1}$$

There are two cases, depending if Γ_i^{rV} is in Γ^V or in $\mathsf{getset}(r)$.

(1) Suppose $i \leq |\Gamma|$ or $i = |\Gamma^r| + 1 = |\Gamma| + 2|r| + 1$. Then $\Gamma_i^{rV} = \Gamma_{i'}^V$ for $i' = \min(i, |\Gamma^r| + 1)$, and we can derive a similar $\longmapsto_{\mathsf{R}}$ transition from (s, Γ^V), using label $\mu' = i', C', \mathtt{cod}(v')$ for some v' and C' such that:
   (a) $r[\Gamma^r] \uplus v[\Gamma^{rVv}] = v'[\Gamma^{Vv'}]$
   (b) $C[\Gamma^{rVv}] = C'[\Gamma^{Vv'}]$

   The premise of (5.1) is hence equal to the premise below, which has however a different conclusion:

$$\frac{(s \uplus v'[\Gamma^{Vv'}]; \Gamma_{i'}^V(C'[\Gamma^{Vv'}])) \longmapsto_{\mathsf{R}} (s'; M')}{(s; \Gamma^V) \xrightarrow{i',C',\mathtt{cod}(v')} (s'; \Gamma^{Vv'}, M')}\ .$$

   We can derive the corresponding transition labelled $i', C', \mathtt{cod}(v')$ from $(u; \Delta, N)$ which will silently reduce to $(u'; \Delta^W)$, then make a visible weak transition to $(u''; \Delta^{Vv'}, N')$ knowing that $(s'; \Gamma^{Vv'}, M')\ \mathcal{S}\ (u''; \Delta^{Wv'}, N')$. We can then replace $\Gamma^{Vv'}$ and $\Delta^{Wv'}$ with Γ^{rVv} and Δ^{rWv}, to prove that $(s'; \Gamma^{rVv}, M')\ \mathcal{S}_1\ (u''; \Delta^{rWv}, N')$, where $\mathcal{S}_1$ is $\mathcal{S}$ where we applied the 'up-to-permutation' technique to move V and W in the middle of v'. This technique is compatible, so $\mathcal{S}_1 \subseteq t_{\mathbf{wp}}(\mathcal{S})$ (and $\mathcal{S}_1 \subseteq t_{\mathbf{sp}}(\mathcal{S})$).

(2) Suppose $i \in \{|\Gamma| + 1, \ldots, |\Gamma| + 2|r|\}$. Then Γ_i^{rV} is either get_ℓ or set_ℓ with $\ell \in \mathsf{dom}(r)$.
   (a) if $\Gamma_i^{rV} = \mathsf{get}_\ell$ then s' is not modified and $M' = r[\Gamma^r]_\ell$ is a context of Γ^r and hence is also a context of $\Gamma^{Vv'}$ using the same v' as above. The result of the transition is:

$$(s \uplus v'[\Gamma^{Vv'}]; \Gamma^{Vv'}, C_1[\Gamma^{Vv'}]) \triangleq x'.$$

   In the weak case, using Lemma 26 we get $(u; \Delta, N) \Longrightarrow (u'; \Delta^W)$, and this term is related to (s, Γ^V) through $\mathbf{wp}(\gtrsim \mathcal{S} \lesssim) \subseteq t_{\mathbf{wp}}(\mathcal{S})$.
   Finally we can relate x' to $(u' \uplus v'[\Delta^{Wv'}]; \Delta^{Wv'}, C_1[\Delta^{Wv'}])$ through $\mathcal{C}(\mathsf{store}(t(\mathcal{S})))$.
   (b) if $\Gamma_i^{rV} = \mathsf{set}_\ell$ then s' is modified at $\ell \in \mathsf{dom}(r)$ (so we only have to change r) and $M' = I = C_1[\Gamma^{Vv'}]$ for $C_1 = I$ (a context with no holes) so the pair progresses again, using the same notations as before, to $\mathcal{C}(\mathsf{store}(t(\mathcal{S})))$.

In summary, we have $\mathsf{store}(\mathcal{R}) \rightsquigarrow \mathsf{store}(\mathcal{S}) \cup t(\mathcal{S}) \cup \mathcal{C}(\mathsf{store}(t(\mathcal{S})))$, and so

$$\mathsf{store} \rightsquigarrow T(\mathsf{store} \cup \mathcal{C}) \tag{5.2}$$

We now establish the progressions for the remaining functions. First, $\mathsf{w} \rightsquigarrow \mathsf{w}$ with the same argument as in the proof of Lemma 19, so $\mathsf{w} \subseteq t$. The most important proof is for $\mathcal{C}$. We assume $(s; \Gamma)\ \mathcal{R}\ (u; \Delta)$, and we analyse the transitions from $(s; \Gamma, C[\Gamma])$.

(1) if $C[\Gamma]$ is a value, then C is a value context C_v, and the same structure as for the corresponding case in Lemma 20 applies here, with the only significant difference being in the third case. The transition of interest is labelled with $i, C_1, \mathtt{cod}(r)$ such that $i \leq |\Gamma| + 1$.

(a) If $i \leq |\Gamma|$, this means the value that is given an argument is one of the Γ_is. Let $i, C_1', \mathtt{cod}(r')$ be the label $i, C_1, \mathtt{cod}(r)$ where we composed the contexts with C_v, so that C_v replaces $[\cdot]_{|\Gamma|+1}$. Using this label on the progression $\mathcal{R} \rightsquigarrow \mathcal{S}$ we obtain a pair in $\mathcal{S}$. We apply first str and then 'up-to-permutation' to add C_v on each side, which puts the desired pair in $t(\mathsf{str}(\mathcal{S}))$.

(b) If $i = |\Gamma| + 1$, and $C_v = [\cdot]_j$, we proceed the same way as above, with the label $j, C_1', \mathtt{cod}(r')$ where C_1' (resp. r') is C_1 (resp. r) where $[\cdot]_j$ replaces all occurrences of $[\cdot]_{|\Gamma|+1}$.

(c) If $i = |\Gamma| + 1$ and C_v is not a hole ($C_v = \lambda x.C_2[x, -]$), then the derivative is $(s \uplus r[\Gamma'^r]; \Gamma'^r, C_3[\Gamma])$ with $\Gamma' = \Gamma, C_v[\Gamma]$ and $C_3 = C_2[C_v, -]$ and with an augmented store, which results in a relation built on $\mathcal{R}$, as follows (we use $\mathcal{C}$ and store, and set $\Delta' = \Delta, C_v[\Delta]$):

$$\dfrac{\dfrac{\dfrac{(s;\Gamma)\ \mathcal{R}\ (u;\Delta)}{(s;\Gamma')\ \mathcal{C}(\mathcal{R})\ (u;\Delta')}}{(s \uplus r[\Gamma'^r];\Gamma'^r)\ \mathsf{store}(\mathcal{C}(\mathcal{R}))\ (u \uplus r[\Delta'^r];\Delta'^r)}}{(s \uplus r[\Gamma'^r];\Gamma'^r, C_2[\Gamma])\ \mathcal{C}(\mathsf{store}(\mathcal{C}(\mathcal{R})))\ (u \uplus r[\Delta'^r];\Delta'^r, C_2[\Gamma])}\ .$$

(2) If $C[\Gamma]$ is not a value, then either $C = E[C_{v1}C_{v2}, -]$ or $C = E[\nu\ell\, C_1[-, \mathsf{get}_\ell, \mathsf{set}_\ell], -]$.

(a) If $C = E[C_{v1}C_{v2}, -]$ and C_{v1} is not a hole, then $C_{v1} = \lambda x.C_1[x, -]$ for some C_1. The transition is of the form $(s;\Gamma, C[\Gamma]) \xrightarrow{\tau} (s;\Gamma, C'[\Gamma])$ with $C' = C_1[C_{v2}, -]$, and similarly for Δ: $(u;\Delta, C[\Delta]) \xrightarrow{\tau} (u;\Delta, C'[\Delta])$. This pair of derivatives is in $\mathcal{C}(\mathcal{R})$.

(b) If if $C = E[C_{v1}C_{v2}, -]$ and $C_{v1} = [\cdot]_i$, then some Γ_i is run, so we also run it starting from the original configuration, with the label $i, C_{v2}, \emptyset$ using the evaluation context function, and therefore progressing to $\mathcal{C}_{\mathsf{e}}(\mathcal{S})$.

(c) The most interesting case is when $C = E[\nu\ell\, C_1[-, \mathsf{get}_\ell, \mathsf{set}_\ell], -]$. Then, $C[\Gamma]$ creates a private location, i.e.,

$$C[\Gamma] = E[\nu\ell\, C_1[\Gamma, \mathsf{get}_\ell, \mathsf{set}_\ell], \Gamma]$$

and $(s;\Gamma, C[\Gamma]) \xrightarrow{\tau} (s \uplus [\ell \mapsto I];\Gamma, C_2[\Gamma, \mathsf{get}_\ell, \mathsf{set}_\ell, \Gamma])$ with $C_2 = E[C_1, -]$. We prove a stronger result, namely that the resulting configurations with the context C_2 are still related if the get_ℓ and set_ℓ operators are available. The derivation is as follows; we use weakening w, exploit $\mathcal{C}$ and store, and write Λ for $\mathsf{get}_\ell, \mathsf{set}_\ell$:

$$\dfrac{\dfrac{\dfrac{(s;\Gamma)\ \mathcal{R}\ (u;\Delta)}{(s \uplus [\ell \mapsto I];\Lambda,\Gamma)\ \mathsf{store}(\mathcal{R})\ (u \uplus [\ell \mapsto I];\Lambda,\Delta)}}{(s \uplus [\ell \mapsto I];\Gamma,\Lambda, C_2[\Gamma,\Lambda])\ \mathcal{C}(\mathsf{store}(\mathcal{R}))\ (u \uplus [\ell \mapsto I];\Delta,\Lambda, C_2[\Delta,\Lambda])}}{(s \uplus [\ell \mapsto I];\Gamma, C_2[\Gamma,\Lambda])\ \mathsf{w}(\mathcal{C}(\mathsf{store}(\mathcal{R})))\ (u \uplus [\ell \mapsto I];\Delta, C_2[\Delta,\Lambda])}\ .$$

To summarise,

$$\mathcal{C}(\mathcal{R}) \rightsquigarrow t(\mathsf{str}(\mathcal{S})) \cup \mathcal{C}(\mathsf{store}(\mathcal{C}(\mathcal{R}))) \cup \mathcal{C}(\mathcal{R}) \cup \mathcal{C}_{\mathsf{e}}(\mathcal{S}) \cup \mathsf{w}(\mathcal{C}(\mathsf{store}(\mathcal{R})))$$

and the right-hand side is included in $T(\mathsf{w} \cup \mathsf{str} \cup \mathcal{C} \cup \mathsf{store} \cup \mathcal{C}_{\mathsf{e}})(\mathcal{R} \cup \mathcal{S})$. Remark that $\mathsf{w} \subseteq t$, $\mathcal{R} \subseteq \mathbf{p}(\mathcal{S})$, and $\mathbf{p} \subseteq t$, so we obtain:

$$\mathcal{C} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathsf{store} \cup \mathcal{C}_{\mathsf{e}})\ . \tag{5.3}$$

We now move on to str and $\mathcal{C}_{\mathsf{e}}$. As in ΛN^1, $\mathsf{v}(\mathcal{R})$ denotes the pairs of value configurations of $\mathcal{R}$, and $\mathsf{n}(\mathcal{R})$ the pairs of non-value configurations of $\mathcal{R}$. It is trivial to check that

$\mathsf{str} \circ \mathsf{v} \subseteq t \circ \mathcal{C}$ and $\mathcal{C}_\mathsf{e} \circ \mathsf{v} \subseteq t \circ \mathcal{C}$, and so by combining with (5.3), both $\mathsf{str} \circ \mathsf{v}$ and $\mathcal{C}_\mathsf{e} \circ \mathsf{v}$ progress to $T(\mathsf{str} \cup \mathcal{C} \cup \mathsf{store} \cup \mathcal{C}_\mathsf{e})$.

It is also straightforward to derive $\mathsf{str} \circ \mathsf{r} \subseteq \mathsf{r} \circ \mathsf{str}$ and $\mathcal{C}_\mathsf{e} \circ \mathsf{r} \subseteq \mathsf{r} \circ \mathcal{C}_\mathsf{e}$, and that $\mathsf{str} \circ \mathsf{n} \rightsquigarrow \mathsf{str}$ and $\mathcal{C}_\mathsf{e} \circ \mathsf{n} \rightsquigarrow \mathcal{C}_\mathsf{e}$. Note that ΛR^1 is quasi-deterministic; indeed, new locations, both for $\nu\ell$ and in the choice of the domain of r in visible transitions, are chosen non-deterministically, but their choice does not matter up to strong bisimilarity. We can now apply Lemma 28 with $f = \mathsf{str}$ and $g = \mathcal{C} \cup \mathsf{store} \cup \mathcal{C}_\mathsf{e}$ to obtain:

$$\mathsf{str} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathsf{store} \cup \mathcal{C}_\mathsf{e}) \tag{5.4}$$

and with $f = \mathcal{C}_\mathsf{e}$ and $g = \mathsf{str} \cup \mathcal{C} \cup \mathsf{store}$ to obtain:

$$\mathcal{C}_\mathsf{e} \rightsquigarrow T(\mathsf{str} \cup \mathcal{C} \cup \mathsf{store} \cup \mathcal{C}_\mathsf{e}) \tag{5.5}$$

Combining (5.2), (5.3), (5.4), and (5.5), yields that $h \triangleq (\mathsf{str} \cup \mathcal{C} \cup \mathsf{store} \cup \mathcal{C}_\mathsf{e})$ progresses to $T(h)$, and hence $h \subseteq t$. □

Having established that $\mathcal{C}$ and $\mathcal{C}_\mathsf{e}$ are below the companion gives as a consequence that our first-order bisimilarity is a congruence under location-free contexts, from which we can derive the soundness implication in Theorem 35 below.

Theorem 35. $(s; M) \equiv (u; N)$ iff $(s; M) \approx (u; N)$.

Proof. **(⇐)** The function $\mathcal{C}_\mathsf{e}$ is below t by Lemma 34. By Lemma 6 we know that $\mathcal{C}_\mathsf{e}(\approx) \subseteq \approx$. In other words $\approx$ is a $\mathcal{C}_\mathsf{e}$-congruence, and in particular $(s; M) \approx (u; N)$ implies $(s; E[M]) \approx (u; E[N])$ for every location-free evaluation context E. This in turn implies that $(s; E[M])$ and $(u; E[N])$ have the same weak visible transitions, which in turn implies that $(s; E[M])\Downarrow$ iff $(u; E[N])\Downarrow$.

(⇒) For completeness, we prove that the following relation $\mathcal{R}$ is a weak bisimulation, where E ranges over location-free evaluation contexts:

$$\mathcal{R} \triangleq \left\{((s; \Gamma, M), (u; \Delta, N)) \text{ s.t. } \forall E\ (s; E[M, \Gamma])\Downarrow \text{ iff } (u; E[N, \Delta])\Downarrow\right\}\ . \tag{5.6}$$

Suppose $(s; \Gamma, M)\ \mathcal{R}\ (u; \Delta, N)$. Since $\mathcal{R}$ is symmetric, we only look at the transitions labelled with μ emanating from $(s; \Gamma, M)$.

When $\mu = \tau$, it holds that $(s; M) \longmapsto_\mathsf{R} (s'; M')$. We then have $(s; E[M, \Gamma])\Downarrow$ iff $(s'; E[M', \Gamma])\Downarrow$ by (quasi-) determinism of $\longmapsto_\mathsf{R}$, so we can conclude $(s'; \Gamma, M')\ \mathcal{R}\ (u; \Delta, N)$ to close the bisimulation diagram.

We now suppose that $\mu \neq \tau$, i.e. M is a value V, $\mu = i, C, \mathsf{cod}(r)$, and $(s; \Gamma, M) \xrightarrow{\mu} (s'; \Gamma'', M')$ for some s', Γ'', M' satisfying

$$\Gamma' = \Gamma, V \quad \Gamma'' = \Gamma', \mathsf{getset}(r) \quad (r[\Gamma''] \uplus s; \Gamma'_i(C[\Gamma''])) \longmapsto_\mathsf{R} (s'; M') \quad (*)$$

Since M is a value, $(s; M) \Downarrow$. By choosing $E = [\cdot]_1$ in (5.6) we know that $(u; N)\Downarrow$ and thus $(u; N) \Longmapsto_\mathsf{R} (u'; W)$ for some value W and store u'. We then obtain the weak transition $(u; \Delta, N) \stackrel{\mu}{\Longrightarrow} (u''; \Delta'', N')$ through $(u'; \Delta, W)$, for some Δ', Δ'', N' such that:

$$\Delta' = \Delta, W \quad \Delta'' = \Delta', \mathsf{getset}(r) \quad (r[\Delta''] \uplus u'; \Delta'_i(C[\Delta''])) \Longmapsto_\mathsf{R} (u''; N') \quad (**)$$

To close the bisimulation diagram, we will now prove that $(s'; \Gamma'', M')\ \mathcal{R}\ (u''; \Delta'', N')$. Let E be a location-free evaluation context, we show that

$$(s'; E[M', \Gamma'']) \Downarrow \quad \text{iff} \quad (u''; E[N', \Delta'']) \Downarrow\ . \tag{5.7}$$

Observe that if $(s_1, M_1) \longmapsto_{\mathsf{R}} (s_1', M_1')$ then $(s_1, E[M_1, \Gamma_1]) \longmapsto_{\mathsf{R}} (s_1', E[M_1', \Gamma_1])$, which implies $(s_1, E[M_1, \Gamma_1])\Downarrow \Leftrightarrow (s_1', E[M_1', \Gamma_1])\Downarrow$ by determinism of $\longmapsto_{\mathsf{R}}$. Using this observation and each reductions in $(*)$ and $(**)$, (5.7) becomes equivalent to:

$$(r[\Gamma''] \uplus s; E[\Gamma_i'(C[\Gamma'']), \Gamma'']) \Downarrow \quad \text{iff} \quad (r[\Delta''] \uplus u'; E[\Delta_i'(C[\Delta'']), \Delta'']) \Downarrow \tag{5.8}$$

We recall that a context of *arity* n is a context with holes $[\cdot]_1, \ldots, [\cdot]_n$ each occurring any number of times and that in an evaluation context the first hole $[\cdot]_1$ is the one that occurs exactly once and in evaluation position. Let F be an evaluation context of arity $|\Gamma| + 1$. Instantiating the definition of $\mathcal{R}$ with F, we have the following equivalence:

$$(s; F[M, \Gamma]) \Downarrow \quad \text{iff} \quad (u; F[N, \Delta]) \Downarrow \tag{5.9}$$

We choose F carefully so that (5.9) is equivalent to (5.8). Let $\ell_i \mapsto C_i$, $i = 1, \ldots, n$, be the collection of location-context pairs of the store context r. Let $C' \triangleq E[[\cdot]_i(C), -]$ i.e. E where the evaluation hole is replaced with the context $[\cdot]_i(C)$. The contexts $C_1, \ldots, C_n$, C, and C', are all of arity $|\Gamma''| = |\Gamma| + 1 + 2n$. For every context D, let $D^\bullet$ be D with the following replacements:

(1) the holes $[\cdot]_{|\Gamma|+1}$ are replaced with x,
(2) the holes $[\cdot]_{|\Gamma|+2i}$ are replaced with get_{ℓ_i},
(3) the holes $[\cdot]_{|\Gamma|+2i+1}$ are replaced with set_{ℓ_i},
(4) all holes $[\cdot]_i$, $i \geq 1$ are simultaneously replaced with $[\cdot]_{i+1}$. This shift leaves $[\cdot]_1$ unused.

Then $C_1^\bullet, \ldots, C_n^\bullet$, $C^\bullet$, $C'^\bullet$ are of arity $|\Gamma| + 1$, with no occurrence of $[\cdot]_1$. We now define the evaluation context F of arity $|\Gamma| + 1$ as follows:

$$F \triangleq \ \mathtt{let}\ x = [\cdot]_1\ \mathtt{in}\ \nu\ell_1 \ldots \nu\ell_n\ \ell_1 := C_1^\bullet;\ \ldots; \ell_n := C_n^\bullet;\ C'^\bullet$$

After $1 + 2n$ steps of reductions (one for the substitution of x with $M = V$, one for each $\nu\ell_i$, one for each assignment) $(s; F[M, \Gamma])$ reduces to $(r[\Gamma''] \uplus s; E[\Gamma_i'(C[\Gamma'']), \Gamma''])$. Similarly $(u; F[N, \Delta])$ first reduces to $(u'; F[W, \Delta])$, and then reduces to $(r[\Delta''] \uplus u'; E[\Delta_i'(C[\Delta'']), \Delta''])$. By determinism of reductions, since each side of (5.9) reduces to the corresponding side of (5.8), we know that (5.9) is equivalent to (5.8). □

Congruence of bisimilarity is restricted either to values ($\mathcal{C}$), or to evaluation contexts ($\mathcal{C}_\mathsf{e}$). It does not hold for arbitrary contexts, but Lemma 38 provides a sufficient condition for some relations between arbitrary terms to be preserved by arbitrary contexts. First we establish weaker results: for evaluation contexts (Lemma 36), then for *non*-evaluation contexts (Lemma 37). Finally Lemma 38 combines the two.

In the following, we use $\asymp$ to denote any of the relations $\sim, \approx$, and $\gtrsim$. (In Lemma 36 F may contain free locations, unlike occurrences in earlier definitions of transitions and of up-to-context functions.)

Lemma 36. Suppose that for all s and Γ, we have $(s; \Gamma, L) \asymp (s; \Gamma, R)$. Then for all s, Γ and evaluation contexts F that may contain free locations, we have $(s; \Gamma, F[L]) \asymp (s; \Gamma, F[R])$.

Proof. Let A be the list of set_ℓ and get_ℓ for all locations ℓ in F. Then we can obtain some location-free F' from F such that $F = F'[-, A]$. By hypothesis we know $(s; \Gamma, A, L) \asymp (s; \Gamma, A, R)$ on which we apply congruence for evaluation contexts $\mathcal{C}_\mathsf{e}$ to derive

$$(s; \Gamma, A, F'[L, A]) \asymp (s; \Gamma, A, F'[R, A])$$

(Lemmas 34 and 6). By weakening we finally obtain $(s; \Gamma, F'[L, A]) \asymp (s; \Gamma, F'[R, A])$. □

Lemma 37. Let L, R be ΛR terms with $(s;\Gamma,L) \asymp (s;\Gamma,R)$ for all environments Γ and stores s. Suppose C is a multi hole context with no hole in evaluation position. Then for all Γ and s we have $(s;\Gamma,C[L]) \asymp (s;\Gamma,C[R])$.

Proof. We do the proof for the most interesting case, $\asymp = \gtrsim$, and we discuss the other cases at the end of the proof.

Let $\mathcal{R}$ relate each configuration $((\ell \mapsto C_v^\ell[L])_\ell; \tilde{C}_v[L], C[L])$ to the one where R replaces L, namely $((\ell \mapsto C_v^\ell[R])_\ell; \tilde{C}_v[R], C[R])$, for all $(C_v^\ell)_\ell$ and $\tilde{C}_v$ families of value contexts (i.e., of the form $\lambda x.C'$), and where C ranges over contexts with no hole in evaluation position. For simplicity we write s_L, s_R, Γ_L, and Γ_R for the corresponding stores and environments. The transitions from both sides, $(s_L;\Gamma_L,C[L])$ and $(s_R;\Gamma_R,C[R])$, have the same shape. We thus show that $\mathcal{R}$ is an expansion up to expansion. We also rely on the fact that L and R are never run.

(1) (Case of silent action.) Since L in $C[L]$ and R in $C[R]$ are not in evaluation position, both sides perform the same kind of transition. The resulting configurations are $(s'_L;\Gamma_L,C_1[L])$ and $(s'_R;\Gamma_R,C_1[R])$ for some C_1. (Even if a set_ℓ or a get_ℓ is involved, and some terms containing L or R are moved to or from the store, the configurations maintain the same shape.)

The only part of the invariant of the relation $\mathcal{R}$ that is not preserved is that L or R may appear in evaluation position, if $C_1[L] = E_1[L,L]$ (where $[\cdot]_1$ is in evaluation position and $[\cdot]_2$ may appear everywhere). In this case, we remark that $F_1 \triangleq E_1[-,L]$ is an evaluation context, on which we can apply Lemma 36 to yield $(s'_L;\Gamma_L,E_1[L,L]) \gtrsim (s'_L;\Gamma_L,E_1[R,L])$. Let $C_2 \triangleq E_1[R,-]$. If C_2 is a context with no hole in evaluation position, we have $(s'_L;\Gamma_L,E[R,L])\ \mathcal{R}\ (s'_R;\Gamma_R,E[R,R])$ and we have closed the diagram. If not, then let E_2 be such that $C_2 = E_2[-,L]$. Applying Lemma 36 as many times as necessary, we can replace occurrences of L with R, one at at time, as long as there remain holes in evaluation position. The progression to $\gtrsim\mathcal{R}$ still holds, since $\gtrsim$ is transitive.

(2) (Case of visible action.) First, since no hole is in evaluation position, $C[L]$ is a value iff $C[R]$ is a value, so they have the same visible actions of the form $i, D, \mathtt{cod}(r)$. We end up with the same shape of configurations we had for the τ transition above, and we therefore proceed similarly.

We have thus proved that $\mathcal{R}$ progresses to $\gtrsim\mathcal{R}$ (expansion up to expansion). In the strong case, we prove that $\mathcal{R}$ progresses to $\sim\mathcal{R}$, and in the weak case we prove that $\mathcal{R}$ weakly progresses to $(\approx\mathcal{R}) \cap (\mathcal{R}\approx)$ (which corresponds to two possible ways of using Lemma 36 in the above proof). Such a refinement is necessary because in the weak case, one can use "up to $\approx$" only when $\approx$ is not on the same side as the challenge. □

Lemma 38. Let $\asymp$ be any of the relations $\sim, \approx$, and $\gtrsim$. Suppose L, R are ΛR terms with $(s;\Gamma,L) \asymp (s;\Gamma,R)$ for all environments Γ and store s. Then also $(s;\Gamma,C[L]) \asymp (s;\Gamma,C[R])$, for every store s, environment Γ and context C.

Proof. Using Lemma 36 and transitivity of $\asymp$, we rewrite the occurrence of L that is in evaluation position into R, and repeat this until there is no such L (such a rewriting may have to be performed more than once if L is not a value but R is so; for example if $L = II$ and $R = I$, then L is in evaluation position in LL on the right and in LR on the left). We finally apply Lemma 37. □

The separation between evaluation contexts and non-evaluation contexts is critical, as handling all contexts together would yield a much larger bisimulation candidate.

Lemma 39. Suppose that E and E' are evaluation contexts and that for all values V and stores s, we have $(s; E[V]) \longmapsto_{\mathsf{R}}^{+} (s; E'[V])$. Then for all environments Γ and stores s, we have $(s; \Gamma, E[M]) \gtrsim (s; \Gamma, E'[M])$.

Proof. For a given Γ we consider $\mathcal{R} = \{(s; \Gamma, E[M]), (s; \Gamma, E'[M]) \mid \text{for all } s \text{ and } M\}$ and the transitions from both sides:

(1) when M is not a value, $(s; M) \longmapsto_{\mathsf{R}} (s'; M')$ and the only transition from both sides is a silent transition, and the derivatives are still in the relation.
(2) when $M = V$ and the challenge transition is from the term on the left-hand side, by hypothesis we have $(s; \Gamma, E[V]) \xrightarrow{\tau}{}^{+} (s; \Gamma, E'[V])$, so the first transition from the left-hand side is a τ. We use up-to-expansion to reach $(s; \Gamma, E'[V])$, which is equal to the right-hand side, and conclude up to reflexivity.
(3) Suppose now $M = V$ and the challenge transition is from the term on the right-hand side. Then the right-hand side makes some transition $(s; \Gamma, E'[V]) \xrightarrow{\alpha} (s'; \Gamma', N')$. We know that $(s; \Gamma, E[V]) \xrightarrow{\tau}{}^{+} (s; \Gamma, E'[V])$ so $(s; \Gamma, E'[V]) \Longrightarrow\!\!\xrightarrow{\alpha} (s'; \Gamma', N')$ and we conclude again up to reflexivity.

We have thus proved that $\mathcal{R}$ is an expansion relation up to expansion and reflexivity. □

Corollary 40. Suppose that E and E' are evaluation contexts and that for every value V and store s, we have $(s; E[V]) \Longmapsto_{\mathsf{R}} (s; E'[V])$. Then for every store s and context C, we have $(s; C[E[M]]) \gtrsim (s; C[E'[M]])$.

Proof. This is a consequence of Lemma 39 and Lemma 38. □

We use Lemma 38 at various places in the example we cover in Section 6. For instance we use it to replace a term $N_1 \triangleq (\lambda x.E[x])M$ (with E an evaluation context) with $N_2 \triangleq E[M]$, under an arbitrary context. Such a property is delicate to prove, even for closed terms, because the evaluation of M could involve reading from a location of the store that itself could contain occurrences of N_1 and N_2.

6. An example

We conclude by discussing an example from [15]. It consists in proving a law between terms of ΛR extended with integers, operators for integer addition and subtraction, and a conditional—those constructs are straightforward to accommodate in the presented framework. For readability, we also use the standard notation for store assignment, dereferencing and sequence: $(\ell := M) \triangleq \mathsf{set}_\ell M$, $!\ell \triangleq \mathsf{get}_\ell I$, and $M; N \triangleq (\lambda x.N)M$ where x does not appear in N. The two terms are the following ones:

- $M \triangleq \lambda g.\nu \ell\ \ell := 0; g(\mathsf{incr}_\ell); \mathtt{if}\ !\ell\ \mathtt{mod}\ 2 = 0\ \mathtt{then}\ I\ \mathtt{else}\ \Omega$
- $N \triangleq \lambda g.g(F); I$,

where $\mathsf{incr}_\ell \triangleq \lambda z.\ell := !\ell + 2$, and $F \triangleq \lambda z.I$. Intuitively, those two terms are weakly bisimilar because the location bound by ℓ in the first term will always contain an even number.

We consider two proofs of the example. In comparison with the proof in [34]: (i) we handle the original example from [15], and (ii) the availability of a broader set of up-to techniques and the possibility of freely combining them allows us to work with smaller relations. In the first proof we work up to the store (through the function store) and up to expansion—two techniques that are not available in [34]. In the second proof we exploit

the up-to-transitivity technique of Section 2, which is only sound for strong bisimilarity, to further reduce the size of the relation we work with.

First proof. We first employ Lemma 38 to reach a variant similar to that of [34]: we make a 'thunk' out of the test in M, and we make N look similar. More precisely, let $\mathsf{test}_\ell \triangleq \lambda z.\mathtt{if}\ !\ell\ \mathtt{mod}\ 2 = 0\ \mathtt{then}\ I\ \mathtt{else}\ \Omega$, we first prove that

- $M \approx M' \triangleq \lambda g.\nu\ell\ \ell := 0; g(\mathsf{incr}_\ell); \mathsf{test}_\ell I$, and
- $N \approx N' \triangleq \lambda g.g(F); FI$.

It then suffices to prove that $M' \approx N'$, which we do using the following relation:

$$\mathcal{R} \triangleq \left\{ \left(s; M', (\mathsf{incr}_\ell, \mathsf{test}_\ell)_{\ell\in\tilde{\ell}}\right), \left(\emptyset; N', (F,F)_{\ell\in\tilde{\ell}}\right) \ \text{ s.t. } \forall \ell \in \tilde{\ell},\ s(\ell) \text{ is even} \right\} .$$

The initial pair of terms is generalised by adding any number of private locations. Indeed M' creates a new location when applied, and its argument can have occurrences of M' that create locations of their own. Relation $\mathcal{R}$ is a weak bisimulation up to store, $\mathcal{C}$ and expansion. We write $(s; \Gamma_{\tilde{\ell}})$ for the left-hand side of a pair in $\mathcal{R}$ and $(\emptyset; \Delta_{\tilde{\ell}})$ for the right-hand side.

Consider a transition $1, C, \mathtt{cod}(r)$ from M' and N'. We write below Γ' for $\Gamma_{\tilde{\ell}}, \mathsf{getset}(r)$ and Δ' for $\Delta_{\tilde{\ell}}, \mathsf{getset}(r)$.

- $(s; \Gamma_{\tilde{\ell}}) \xrightarrow{1, C, \mathtt{cod}(r)} (s \uplus r[\Gamma']; \Gamma', \nu\ell\ \ell := 0; C[\Gamma'](\mathsf{incr}_\ell); \mathsf{test}_\ell I)$
- $(\emptyset; \Delta_{\tilde{\ell}}) \xrightarrow{1, C, \mathtt{cod}(r)} (r[\Delta']; \Delta', C[\Delta'](F); FI)$

In the first line, we make the configuration run two τ transitions, so that $\nu\ell$ and $\ell := 0$ get executed. Now we have a new store $s' = s \uplus (\ell \mapsto 0)$ (as $s'(\ell)$ is even, we remain within the bisimulation candidate).

Now the main term is $C[\Gamma'](\mathsf{incr}_\ell); \mathsf{test}_\ell I)$, which can be rewritten to $D[\Gamma_{\tilde{\ell}.\ell}, \mathsf{getset}(r)]$ for some context D. On the right-hand side $C[\Delta'](F); FI$ can be rewritten to $D[\Delta_{\tilde{\ell}.\ell}, \mathsf{getset}(r)]$ as well. By construction $(s'; \Gamma_{\tilde{\ell}.\ell})\ \mathcal{R}\ (\emptyset; \Delta_{\tilde{\ell}.\ell})$ hence

$$(s' \uplus r[\Gamma']; \Gamma_{\tilde{\ell}.\ell}, \mathsf{getset}(r)) \quad \mathsf{store}(\mathcal{R}) \quad (r[\Delta']; \Delta_{\tilde{\ell}.\ell}, \mathsf{getset}(r)) .$$

Now we first apply $\mathcal{C}$ with context D, then weakening w to remove incr_ℓ and test_ℓ which do not appear outside of D, and we thus obtain the required pair.

Having handled M' and N', we look at a transition $i, C, \mathtt{cod}(r)$ coming from some incr_ℓ (and F on the other side). On the right-hand side, $(\emptyset \uplus r[\Delta']; F(C[\Delta']))$ immediately $\longmapsto_{\mathsf{R}}$-reduces to $(\emptyset \uplus r[\Delta']; I)$, discarding its argument. On the left-hand side, a few more steps are necessary to reach I: $(s \uplus r[\Gamma']; \mathsf{incr}_\ell(C[\Gamma']))$ also discards its argument to become first $(s \uplus r[\Gamma']; \mathsf{set}_\ell(\mathsf{get}_\ell + 2))$ and then, after three steps, $(s' \uplus r[\Gamma']; I)$, with $s' \triangleq s[\ell \mapsto s(\ell) + 2]$. Since $(s'; \Gamma_{\tilde{\ell}})\ \mathcal{R}\ (\emptyset; \Delta_{\tilde{\ell}})$, it is enough to apply up to store (adding $r[\Gamma']$), $\mathcal{C}$ (adding I), and expansion (accounting for the extra τ steps of the left-hand side), to conclude. Adapting the terms accordingly, the reasoning for test_ℓ is the same: $(s \uplus r[\Gamma']; \mathsf{test}_\ell(C[\Gamma']))$ reduces to $(s \uplus r[\Gamma']; I)$ since $s(\ell)$ is even.

Second proof. We first preprocess the terms using Lemma 38, to add a few artificial internal steps to N, so that we can carry out the reminder of the proof using strong bisimilarity, which enjoys more up-to techniques than weak bisimilarity:

- $M \approx M' \triangleq \lambda g.\nu\ell\ \ell := 0; g(\mathsf{incr}_\ell); \mathsf{test}_\ell I$,
- $N \approx N'' \triangleq \lambda g.I; I; g(\mathsf{incr}_0); \mathsf{test}_0 I$.

where incr_0 and test_0 are pure functions that return I on any input, taking the same number of internal steps as incr_ℓ and test_ℓ. We show that $M' \sim N''$ by proving that the following relation $\mathcal{S}$ is a strong bisimulation *up to unfolding, store, weakening, strengthening, transitivity and context* (a technique unsound in the weak case):

$$\mathcal{S} \triangleq \{((\emptyset; M'), (\emptyset; N''))\} \ \cup \ \{((\ell \mapsto 2n; \mathsf{incr}_\ell, \mathsf{test}_\ell), (\emptyset; \mathsf{incr}_0, \mathsf{test}_0)) \ \forall\ n \in \mathbb{N}\}$$

This relation uses only one location; it is the union of the singleton relation $\{((\emptyset; M'), (\emptyset; N''))\}$ and the relation relating $(\ell \mapsto 2n; \mathsf{incr}_\ell, \mathsf{test}_\ell)$ to $(\emptyset; \mathsf{incr}_0, \mathsf{test}_0)$ for every integer that can be stored at that location. In the diagram-chasing arguments for $\mathcal{S}$, essentially a pair of derivatives is proved to be related under the function

$$\mathbf{sp} \circ \mathbf{sp} \circ \mathsf{star} \circ (\mathsf{str} \cup \mathsf{store} \cup \mathcal{C} \cup \mathsf{w})^\omega$$

where $\mathsf{star} : \mathcal{R} \mapsto \mathcal{R}^\star$ is the reflexive-transitive closure function.

This up-to technique, unsound in the weak case (transitivity is unsound), is powerful enough to make the bisimulation considerably smaller. Proving that the second member of $\mathcal{S}$ progresses to itself (up to store) is straightforward. We focus on the following transitions from M' and N'':

$$(\emptyset, M') \xrightarrow{1, C, \mathsf{cod}(r)} (\ r[\Gamma];\ \Gamma, \nu\ell\ \ell := 0; C[\Gamma]\,(\mathsf{incr}_\ell); \mathsf{test}_\ell I\ ) \triangleq H_1$$
$$(\emptyset, N'') \xrightarrow{1, C, \mathsf{cod}(r)} (\ r[\Delta];\ \Delta, I; I; \qquad C[\Delta](\mathsf{incr}_0); \mathsf{test}_0 I\ ) \triangleq H_2$$

where $\Gamma = M', \mathsf{getset}(r)$ and $\Delta = N'', \mathsf{getset}(r)$. We use $\mathbf{sp}$ as an up-to technique[2] twice so to run two steps of reduction on both sides:

$$H_1 \xrightarrow{\tau}\xrightarrow{\tau} H_1' \qquad \text{and} \qquad H_2 \xrightarrow{\tau}\xrightarrow{\tau} H_2' \ .$$

This way we trigger $\nu\ell$ and $\ell := 0$ and obtain two configurations H_1' and H_2' that can be related using a few up-to functions:

$$(\ r[\Gamma] \uplus (\ell \mapsto 0);\ \Gamma,\ C[\Gamma](\mathsf{incr}_\ell); \mathsf{test}_\ell I\ ) = H_1' \tag{6.1}$$
$$\mathsf{w}(\mathcal{C}(\mathsf{store}(\mathsf{str}(\mathcal{S})))) \quad (\ r[\Gamma];\ \Gamma,\ C[\Gamma](\mathsf{incr}_0); \mathsf{test}_0 I\ ) \tag{6.2}$$
$$\mathcal{C}(\mathsf{store}(\mathcal{S})) \quad (\ r[\Delta];\ \Delta,\ C[\Delta](\mathsf{incr}_0); \mathsf{test}_0 I\ ) = H_2' \ . \tag{6.3}$$

We detail below how we go from (6.1) to (6.2). We write $\Gamma_\ell \triangleq \mathsf{incr}_\ell, \mathsf{test}_\ell$ and $\Gamma_0 \triangleq \mathsf{incr}_0, \mathsf{test}_0$, and use $-$ as a shorthand for the relation mentioned in the line above it:

$(\ell \mapsto 0; \Gamma_\ell)$	$\mathcal{S}$	$(\emptyset; \Gamma_0)$
$(\ell \mapsto 0; \Gamma_\ell, M')$	$\mathsf{str}(-)$	$(\emptyset; \Gamma_0, M')$
$(r[\Gamma] \uplus \ell \mapsto 0; \Gamma_\ell, \Gamma)$	$\mathsf{store}(-)$	$(r[\Gamma]; \Gamma_0, \Gamma)$
$(r[\Gamma] \uplus \ell \mapsto 0; \Gamma_\ell, \Gamma, C[\Gamma](\mathsf{incr}_\ell); \mathsf{test}_\ell I))$	$\mathcal{C}(-)$	$(r[\Gamma]; \Gamma_0, \Gamma, C[\Gamma](\mathsf{incr}_0); \mathsf{test}_0 I)$
$(r[\Gamma] \uplus \ell \mapsto 0; \quad \Gamma, C[\Gamma](\mathsf{incr}_\ell); \mathsf{test}_\ell I))$	$\mathsf{w}(-)$	$(r[\Gamma]; \quad \Gamma, C[\Gamma](\mathsf{incr}_0); \mathsf{test}_0 I)$

Going from (6.2) to (6.3) is easier:

$(\emptyset; M')$	$\mathcal{S}$	$(\emptyset; N'')$
$(r[\Gamma]; \Gamma)$	$\mathsf{store}(-)$	$(r[\Delta]; \Delta)$
$(r[\Gamma]; \Gamma, C[\Gamma](\mathsf{incr}_0); \mathsf{test}_0 I)$	$\mathcal{C}(-)$	$(r[\Delta]; \Delta, C[\Delta](\mathsf{incr}_0); \mathsf{test}_0 I)$

We have thus proved that $H_1\ f(\mathcal{S})\ H_2$ where $f = \mathbf{sp} \circ \mathbf{sp} \circ \mathsf{star} \circ (\mathsf{str} \cup \mathsf{store} \cup \mathcal{C} \cup \mathsf{w})^\omega$ is below t, and hence $\mathcal{S} \rightsquigarrow f(\mathcal{S}) \cup \mathsf{store}(\mathcal{S})$. To conclude, $\mathcal{S}$, as a strong bisimulation up to (unfolding, store, weakening, strengthening, transitivity and context), is included in $\sim$.

[2] If $\xrightarrow{\tau}$ is deterministic then $(\xrightarrow{\tau} \mathcal{R} \xleftarrow{\tau}) \subseteq \mathbf{sp}(\mathcal{R})$.

The difference between the relation $\mathcal{R}$ in the first proof and the proofs in [15, 34] is that $\mathcal{R}$ only requires locations that appear free in the tested terms; in contrast, the relations in [15, 34] need to be closed under all possible extensions of the store, including extensions in which related locations are mapped onto arbitrary context-closures of related values. We avoid this thanks to the up-to-store function, by discarding these extensions immediately after their introduction. The reason why, both in [15, 34] and in the first proof above, several locations have to be considered is that, with bisimulations akin to environmental bisimulation, the input for a function is built using the values that occur in the candidate relation. In our example, this means that the input for a function can be a context-closure of M and N; hence uses of the input may cause several evaluations of M and N, each of which generates a new location. In this respect, it is surprising that our second proof avoids multiple allocations (the candidate relation $\mathcal{S}$ only mentions one location). This is due to the massive combination of up-to techniques whereby, whenever a new location is created, a double application of up-to-context (the 'double' is obtained from up-to-transitivity) together with some administrative work (given by the other techniques) allows us to absorb the location.

7. Conclusions

In this paper we have studied how to transport the rich theory of 'up-to' techniques that exists for plain (first-order) LTSs and bisimilarity, and rooted in fixed-point theory, onto languages whose LTS and bisimilarity go beyond the first-order format. For this we have considered the π-calculus, the pure call-by-name λ-calculus, and a call-by-value λ-calculus extended with imperative features.

The approach that we have proposed exhibits fully abstract translations of the LTSs and bisimilarities of these languages onto first-order LTSs. In this way, one can directly reuse the large corpus of up-to techniques that are available on first-order LTSs. The only exception to this regards basic up-to techniques that are specific to the new languages, such as up-to-context. Most important, the approach allows one to take arbitrarily complex combinations of up-to techniques, whose soundness is guaranteed by those of the corresponding first-order techniques. Direct proofs of such combinations, on the source languages, can be long and delicate. We have given examples of uses of such combinations. In particular, the second proof of the example dealt with in Section 6 is, in our opinion, a striking example of the benefits of the up-to techniques. It is hard to imagine how to the example could be handled without up-to techniques; compared to similar proofs in the literature and discussed in that section — all of which make use of some forms of up-to techniques — the relation employed and the proof work needed have been significantly reduced due to the large set of up-to techniques referred to.

The work in this paper can be a further motivation for the development of a comprehensive formalised library of up-to techniques for first-order LTSs. Using the approach proposed in the paper the library could then be applied to a wide range of languages. Other directions for future work include testing the approach on other languages, for instance languages for mobility with explicit notions of location (see [7] for a survey), or testing it on other forms of bisimulation (e.g., open bisimulation). By the time the revision of this paper has been completed, one such application of the approach has been made, for a calculus with delimited-control operators with dynamic prompt generation [3].

We would also like to see if the approach proposed in this paper, based on translations to first-order models, could be adapted to handle the theory of unique solutions of equations and contractions [37, 9], which allows one to implicitly use up-to techniques, with the goal of avoiding the development of theories of equations or contractions that are specific to a particular language or bisimulation.

Acknowledgements

We are delighted to be able to contribute to the Festschrift in honour of Jos Baeten. We would like to take this opportunity for heartily thanking him for having been such an inspiring figure, both for all his many technical and scientific contributions and for his work in favour of the concurrency theory community.

We would like also to thanks the anonymous referees for many useful comments. Pous was supported by the European Research Council (ERC) under the European Union's Horizon 2020 programme (CoVeCe, grant agreement No 678157). Sangiorgi acknowledges support from the MIUR-PRIN project 'Analysis of Program Analyses' (ASPRA, ID: 201784YSZ5_004), and from the European Research Council (ERC) Grant DLV-818616 DIAPASoN.

References

[1] M. Abadi and A.D. Gordon. A bisimulation method for cryptographic protocols. In Chris Hankin, editor, *ESOP'98*, volume 1381 of *LNCS*, pages 12–26. Springer, 1998.

[2] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley, 1989.

[3] Andrés Aristizábal, Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. Environmental Bisimulations for Delimited-Control Operators with Dynamic Prompt Generation. In *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, volume 52 of *LIPIcs*, Porto, Portugal, June 2016.

[4] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29:737–760, 1992.

[5] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. A complete normal-form bisimilarity for state. In Mikolaj Bojanczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*, pages 98–114. Springer, 2019.

[6] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. A complete normal-form bisimilarity for algebraic effects and handlers. In Zena M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*, volume 167 of *LIPIcs*, pages 7:1–7:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[7] Ilaria Castellani. Process algebras with localities. In A. Ponse J. Bergstra and S. Smolka, editors, *Handbook of Process Algebra*, pages 945–1045. Elsevier, 2001.

[8] K. Chaudhuri, M. Cimini, and D. Miller. Formalization of the bisimulation-up-to technique and its meta theory. Draft, 2014.

[9] Adrien Durier, Daniel Hirschkoff, and Davide Sangiorgi. Divergence and unique solution of equations. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory, CONCUR 2017*, volume 85 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[10] D. Hirschkoff. A full formalisation of pi-calculus theory in the calculus of constructions. In *TPHOLs*, volume 1275 of *LNCS*, pages 153–169. Springer, 1997.

[11] C.-K. Hur, G. Neis, D. Dreyer, and V. Vafeiadis. The power of parameterization in coinductive proof. In *POPL*, pages 193–206. ACM, 2013.

[12] A. Jeffrey and J. Rathke. Towards a theory of bisimulation for local names. In *LICS*, pages 56–66, 1999.
[13] V. Koutavas and M. Hennessy. First-order reasoning for higher-order concurrency. *Computer Languages, Systems & Structures*, 38(3):242–277, 2012.
[14] V. Koutavas, P. B. Levy, and E. Sumii. From applicative to environmental bisimulation. *Electr. Notes Theor. Comput. Sci.*, 276:215–235, 2011.
[15] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In *POPL'06*, pages 141–152. ACM, 2006.
[16] S.B. Lassen. Relational reasoning about contexts. In *Higher-order operational techniques in semantics*, pages 91–135. Cambridge University Press, 1998.
[17] S.B. Lassen. *Relational Reasoning about Functions and Nondeterminism*. PhD thesis, Department of Computer Science, University of Aarhus, 1998.
[18] S.B. Lassen. Bisimulation in untyped lambda calculus: Böhm trees and bisimulation up to context. *Electr. Notes Theor. Comput. Sci.*, 20:346–374, 1999.
[19] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In Catuscia Palamidessi, editor, *CONCUR 2000 — Concurrency Theory*, pages 243–258, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
[20] M. Lenisa. *Themes in Final Semantics*. Ph.D. thesis, Università di Pisa, 1998.
[21] Jean-Marie Madiot, Damien Pous, and Davide Sangiorgi. Bisimulations up-to: Beyond first-order transition systems. In Paolo Baldan and Daniele Gorla, editors, *Proc. CONCUR 2014*, volume 8704 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 2014.
[22] M. Merro and F. Zappa Nardelli. Behavioral theory for mobile ambients. *J. ACM*, 52(6):961–1023, 2005.
[23] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
[24] J.Å. Pohjola and J. Parrow. Bisimulation up-to techniques for psi-calculi. Draft, 2014.
[25] D. Pous. *Techniques modulo pour les bisimulations*. Phd thesis, École Normale Supérieure de Lyon, February 2008.
[26] D. Pous and D. Sangiorgi. Enhancements of the bisimulation proof method. In *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2012.
[27] Damien Pous. Complete lattices and up-to techniques. In *APLAS*, volume 4807 of *Lecture Notes in Computer Science*, pages 351–366. Springer Verlag, 2007.
[28] Damien Pous. Coinduction all the way up. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 307–316. ACM, 2016.
[29] Damien Pous and Davide Sangiorgi. Bisimulation and coinduction enhancements: A historical perspective. *Formal Asp. Comput.*, 31(6):733–749, 2019.
[30] J. Rot, M. Bonsangue, and J. Rutten. Coalgebraic bisimulation-up-to. In *SOFSEM'13*, volume 7741 of *LNCS*, pages 369–381. Springer, 2013.
[31] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Inf. and Comp.*, 111(1):120–153, 1994.
[32] D. Sangiorgi. Locality and true-concurrency in calculi for mobile processes. In *TACS'94*, volume 789 of *Lecture Notes in Computer Science*, pages 405–424. Springer, 1994.
[33] D. Sangiorgi. On the bisimulation proof method. *J. of MSCS*, 8:447–479, 1998. A short version in Proc. MFCS'95, 1995.
[34] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. *ACM Trans. Program. Lang. Syst.*, 33(1):5, 2011.
[35] D. Sangiorgi and R. Milner. The problem of "Weak Bisimulation up to". In W.R. Cleveland, editor, *Proc. CONCUR '92*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer Verlag, 1992.
[36] D. Sangiorgi and D. Walker. *The Pi-Calculus: a theory of mobile processes*. Cambridge University Press, 2001.
[37] Davide Sangiorgi. Equations, contractions, and unique solutions. *ACM Trans. Comput. Log.*, 18(1):4:1–4:30, 2017.
[38] Jaroslav Sevcík, Viktor Vafeiadis, Francesco Zappa Nardelli, Suresh Jagannathan, and Peter Sewell. Compcerttso: A verified compiler for relaxed-memory concurrency. *J. ACM*, 60(3):22:1–22:50, 2013.
[39] E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theor. Comput. Sci.*, 375(1-3):169–192, 2007.

[40] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *J. ACM*, 54(5), 2007.
[41] N.D. Turner. *The polymorphic pi-calculus: Theory and Implementation*. PhD thesis, Department of Computer Science, University of Edinburgh, 1996.