Bisimulations Up-to: Beyond First-Order Transition Systems

Jean-Marie Madiot[1], Damien Pous[1], and Davide Sangiorgi[2]

[1] ENS Lyon, Université de Lyon, CNRS, INRIA, France
[2] Università di Bologna, INRIA, Italy

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 the notion of compatible function for fixed-point theory.

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'* [14,19,20]. 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., [23,16,11]).

Several forms of bisimulation enhancements have been introduced: 'bisimulation up-to bisimilarity' [17] 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; 'bisimulation up-to-context' [21], 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 [7,23], techniques for shrinking or enlarging the environment in languages with information hiding mechanisms (e.g., existential types, encryption and decryption constructs [1,25,24]), frame equivalence in the psi-calculi [18], or higher-order languages [12,10]. Lastly, it is important to notice 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. And 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, in certain models up-to bisimilarity may fail for weak bisimilarity, and in certain languages up-to bisimilarity and context may fail even if bisimilarity is a congruence relation and is strong (treating internal moves as any other move).

This problem has been the motivation for the development of a theory of enhancements, summarised in [19]. 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. 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 [5,4], the psi-calculi [18], or a miniML language [6].

We consider three languages: the π-calculus, the call-by-name λ-calculus, and an imperative call-by-value λ-calculus. Other calculi like the Higher-Order π-calculus can be handled in a similar way; we omit the details here for lack of space. We moreover focus on weak bisimilarity, since its theory is more delicate than that of strong bisimilarity. When we translate a transition system into a first-order one, the grammar for the labels can be complex (e.g. include terms, labels, or contexts). What makes the system 'first-order' is that labels are taken as syntactic atomic objects, that may only be checked for syntactic equality. Note that full abstraction of the translation does not imply that the up-to techniques come for free: further conditions must be ensured. We shall see this with the π-calculus, where early bisimilarity can be handled but not the late one.

Forms of up-to context have already been derived for the languages we consider in this paper [11,23,22]. 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, substitutivity (or congruence) 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., [26]) in which the issue of the choice of fresh names is resolved by ordering the names. 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 [8]. 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 LTSs from [8] would lead to technical problems similar to those discussed in Remark 2. In the λ-calculus, limited forms of up-to techniques have been developed for applicative bisimilarity, where the soundness of the up-to context has still open problems [12,11]. More powerful versions of up-to context exist for forms of bisimilarity on open terms (e.g., open bisimilarity or head-normal-form bisimilarity) [13]. Currently, the form of bisimilarity for closed higher-order terms that allows the richest range of up-to techniques is environmental bisimilarity [22,9]. 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.

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.

2 First-Order Bisimulation and Up-to Techniques

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. *Suppose $\mathcal{R}, \mathcal{S}$ are relations on the processes of an LTS. Then $\mathcal{R}$* strongly progresses to *$\mathcal{S}$, written $\mathcal{R} \rightsquigarrow_{\mathbf{sp}} \mathcal{S}$, if $\mathcal{R} \subseteq \mathcal{S}$ and if $P \mathrel{\mathcal{R}} Q$ implies:*

- *whenever $P \xrightarrow{\mu} P'$ there is Q' s.t. $Q \xrightarrow{\mu} Q'$ and $P' \mathrel{\mathcal{S}} Q'$;*
- *whenever $Q \xrightarrow{\mu} Q'$ there is P' s.t. $P \xrightarrow{\mu} P'$ and $P' \mathrel{\mathcal{S}} Q'$.*

A relation $\mathcal{R}$ is a strong bisimulation *if $\mathcal{R} \rightsquigarrow_{\mathbf{sp}} \mathcal{R}$; 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 $\overset{\hat{\mu}}{\Longrightarrow}$ is $\Longrightarrow\xrightarrow{\hat{\mu}}\Longrightarrow$ where $\Longrightarrow$ is the reflexive transitive closure of $\xrightarrow{\tau}$. *Weak progression*, $\mathcal{R} \rightsquigarrow_{\mathbf{wp}} \mathcal{S}$, and *weak bisimilarity*, $\approx$, are obtained from Definition 1 by allowing the processes to answer using $\overset{\hat{\mu}}{\Longrightarrow}$ rather than $\xrightarrow{\mu}$.

Below we summarise the ingredients of the theory of bisimulation enhancements for first-order LTSs from [19] that will be needed in the sequel. We use f and g to range over 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 2. *A function f is* sound for $\sim$ *if $\mathcal{R} \rightsquigarrow_{\mathbf{sp}} f(\mathcal{R})$ implies $\mathcal{R} \subseteq \sim$, for all $\mathcal{R}$; similarly, f is* sound for $\approx$ *if $\mathcal{R} \rightsquigarrow_{\mathbf{wp}} f(\mathcal{R})$ implies $\mathcal{R} \subseteq \approx$, for all $\mathcal{R}$.*

Unfortunately, the class of sound functions does not enjoy good algebraic properties. 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 3. *We write $f \rightsquigarrow_{\mathbf{p}} g$ when $\mathcal{R} \rightsquigarrow_{\mathbf{p}} \mathcal{S}$ implies $f(\mathcal{R}) \rightsquigarrow_{\mathbf{p}} g(\mathcal{S})$ for all $\mathcal{R}$ and $\mathcal{S}$. A monotone function f on relations is* $\mathbf{p}$-compatible *if $f \rightsquigarrow_{\mathbf{p}} f$.*

In other terms [19], f is $\mathbf{p}$-compatible iff $f \circ \mathbf{p} \subseteq \mathbf{p} \circ f$ where $\mathbf{p}(\mathcal{S})$ is the union of all $\mathcal{R}$ such that $\mathcal{R} \rightsquigarrow_{\mathbf{p}} \mathcal{S}$ and $\circ$ denotes function composition. Note that $\mathcal{R} \rightsquigarrow_{\mathbf{p}} \mathcal{S}$ is equivalent to $\mathcal{R} \subseteq \mathbf{p}(\mathcal{S})$.

Lemma 1. *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 and the function mapping any relation onto bisimilarity (for the strong or weak case, respectively). 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 omega-iteration (where the omega-iteration f^ω of a function f maps $\mathcal{R}$ to $\bigcup_{n \in \mathbb{N}} f^n(\mathcal{R})$).

Other examples of compatible functions are typically contextual closure functions, mapping a relation into its closure w.r.t. a given set of contexts. For such functions, the following lemma shows that the compatibility of up-to-context implies substitutivity of (strong or weak) bisimilarity.

Lemma 2. *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 relational composition:

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

where $f(\mathcal{R})\; g(\mathcal{R})$ indicates the composition of the two relations $f(\mathcal{R})$ and $g(\mathcal{R})$. Using chaining we can obtain the compatibility of the function 'up to transitivity' mapping any relation $\mathcal{R}$ onto its reflexive and transitive closure $\mathcal{R}^\star$. Another example of **sp**-compatible function is 'up to bisimilarity' ($\mathcal{R} \mapsto {\sim}\mathcal{R}{\sim}$).

In contrast, in the weak case bisimulation up to bisimilarity 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 $\gtrsim$ [3]. 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); see [15] for its definition. Up-to-expansion yields a function ($\mathcal{R} \mapsto {\gtrsim}\mathcal{R}{\lesssim}$) that is **wp**-compatible. As a consequence, the same holds for the 'up-to expansion and contexts' function. 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 [19].

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 by **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.

Last, note that to use a function f in combinations of up-to techniques, it is actually not necessary that f be **p**-compatible: for example proving that f progresses to $f \cup g$ and g progresses to g is enough, as then $f \cup g$ would be compatible. Extending this reasoning allows us to make use of 'second-order up-to techniques' to reason about compatibility of functions. When F is a set of functions, we say that F is **p**-*compatible up to* if for all f in F, it holds that $f \rightsquigarrow_{\mathbf{p}} (g \cup (\cup F))^\omega$ for a function g that has already been proven compatible. (We sometimes say that F is **p**-*compatible up to* g, to specify which compatible function is employed.) Lemma 1 and 2 remain valid when 'f is compatible' is replaced by '$f \in F$ and F is compatible up to'.

Terminology We will simply say that a function is *compatible* to mean that it is both **sp**-compatible and **wp**-compatible; similarly for compatibility up to. In languages defined from a grammar, a context C is a term with numbered holes $[\cdot]_1, \dots, [\cdot]_n$, and each hole $[\cdot]_i$ can appear any number of times in C.

3 The π-calculus

The syntax and operational semantics of the π-calculus are recalled in [15]. We consider the early transition system, in which transitions are of the forms

$$P \overset{ab}{\longmapsto}_\pi P' \qquad P \overset{\overline{a}b}{\longmapsto}_\pi P' \qquad P \overset{\overline{a}(b)}{\longmapsto}_\pi P' \ .$$

In the third transition, called bound output transition, name b is a binder for the free occurrences of b in P' and, as such, it is subject to α-conversion. The definition of bisimilarity takes α-conversion into account. The clause for bound output of strong early bisimilarity says (fn(Q) indicates the names free in Q):

– 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 $P' \sim Q'$.

(The complete definition of bisimilarity is recalled in [15]). When translating the π-calculus semantics to a first-order one, α-conversion and the condition $b \notin \mathrm{fn}(Q)$ have to be removed. To this end, one has to force an agreement between two bisimilar process 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 LTS, 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 objects (without α-conversion). We write $\sim^{\mathrm{e}}$ and $\approx^{\mathrm{e}}$ for strong and weak early bisimilarity of the π-calculus.

Theorem 1. *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 $(c, P) \approx (c, Q)$.*

The above full abstraction result allows us to import the theory of up-to techniques for first-order LTSs and bisimilarity, both in the strong and the weak case. We have however to prove the soundness of up-to techniques that are specific to the π-calculus. Function isub implements 'up to injective name substitutions':

$$\mathsf{isub}(\mathcal{R}) \triangleq \{((d, P\sigma), (d, Q\sigma)) \;\text{ s.t. } (c, P) \;\mathcal{R}\; (c, Q),\, \mathrm{fn}(P\sigma) \cup \mathrm{fn}(Q\sigma) \leq d, \\ \text{and } \sigma \text{ is injective on } \mathrm{fn}(P) \cup \mathrm{fn}(Q) \;\; \} \quad .$$

A subtle drawback is the need of another function manipulating names, str, allowing us to replace the index c in a named process (c, P) with a lower one:

$$\mathsf{str}(\mathcal{R}) \triangleq \{((d, P), (d, Q)) \;\text{ s.t. } (c, P) \;\mathcal{R}\; (c, Q) \text{ and } \mathrm{fn}(P, Q) \leq d \;\} \;\; .$$

Lemma 3. *The set $\{\mathsf{isub}, \mathsf{str}\}$ is compatible up to.*

The up-to-context function is decomposed into a set of smaller context functions, called *initial* [19], 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 returning the closure of a relation under the operators of output prefix, restriction, replication, parallel composition, and sum.

Definition 4. *If $\mathcal{R}$ is a relation on π^1, we define $\mathcal{C}_o(\mathcal{R})$, $\mathcal{C}_\nu(\mathcal{R})$, $\mathcal{C}_!(\mathcal{R})$, $\mathcal{C}_|(\mathcal{R})$ and $\mathcal{C}_+(\mathcal{R})$ by saying that whenever $(c, P) \;\mathcal{R}\; (c, Q)$,*

- $(c, \overline{a}b.P) \;\; \mathcal{C}_o(\mathcal{R}) \;\; (c, \overline{a}b.Q)$, *for any a, b with $a, b \leq c$,*
- $(c, \nu a.P) \;\; \mathcal{C}_\nu(\mathcal{R}) \;\; (c, \nu a.Q)$,
- $(c, !P) \;\; \mathcal{C}_!(\mathcal{R}) \;\; (c, !Q)$;

and, whenever $(c, P_1) \;\mathcal{R}\; (c, Q_1)$ and $(c, P_2) \;\mathcal{R}\; (c, Q_2)$,

- $(c, P_1 \mid Q_1) \;\; \mathcal{C}_|(\mathcal{R}) \;\; (c, P_2 \mid Q_2)$,
- $(c, P_1 + Q_1) \;\; \mathcal{C}_+(\mathcal{R}) \;\; (c, P_2 + Q_2)$.

While bisimilarity in the π-calculus is not preserved by input prefix, a weaker rule holds (where $=$ can be $\sim^{\mathrm{e}}$ or $\approx^{\mathrm{e}}$):

$$\frac{P = Q \quad \text{and } P\{c/b\} = Q\{c/b\} \text{ for each } c \text{ free in } P, Q}{a(b).P = a(b).Q} \tag{1}$$

We define $\mathcal{C}_i$, the function for input prefix, accordingly: we have $(d, a(b).P) \;\mathcal{C}_i(\mathcal{R})\; (d, a(b).Q)$ if $a \leq d$ and $(d+1, P\{c/b\}) \;\mathcal{R}\; (d+1, Q\{c/b\})$ for all $c \leq d+1$.

Theorem 2. *The set $\{\mathcal{C}_o, \mathcal{C}_i, \mathcal{C}_\nu, \mathcal{C}_!, \mathcal{C}_|, \mathcal{C}_+\}$ is* **sp**-*compatible up to* $\mathsf{isub} \cup \mathsf{str}$.

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 3. *The set* $\{\mathcal{C}_o, \mathcal{C}_i, \mathcal{C}_\nu, \mathcal{C}_!, \mathcal{C}_|, \mathcal{C}_{g+}\}$ *is* **wp**-*compatible up to* $\mathsf{isub} \cup \mathsf{str} \cup \mathbf{b}$ *where* $\mathbf{b} = (\mathcal{R} \mapsto {\sim}\mathcal{R}{\sim})$ *is 'up to bisimilarity'.*

The compatibility of these functions is not a logical consequence of the up to context results in the π-calculus; instead we prove them from scratch [15], with the benefit of having a separate proof for each initial context.

As a byproduct of the compatibility of these initial context functions, and using Lemma 2, we derive the standard substitutivity properties of strong and weak early bisimilarity, including the rule (1) for input prefix.

Corollary 1. *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 (1) is valid both for $\sim^{\mathrm{e}}$ and $\approx^{\mathrm{e}}$.*

Remark 1. Late bisimilarity makes use of transitions $P \xmapsto{a(b)}_{\pi} P'$ where b is bound, the definition of bisimulation containing a quantification over names. To capture this bisimilarity in a first-order LTS we would need to have two transitions for the input $a(b)$: one to fire the input a, leaving b uninstantiated, and another to instantiate 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.

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 standard syntax of λ-terms, and the rules for call-by-name reduction, are recalled in [15]. 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; extension to open terms is made using closing abstractions (i.e., abstracting on all free variables). The reduction relation of ΛN is $\longmapsto_{\mathsf{n}}$, and $\Longmapsto_{\mathsf{n}}$ its reflexive and transitive closure.

As bisimilarity for the λ-calculus we consider *environmental bisimilarity* [22,9], which allows a set of up-to techniques richer than Abramsky's applicative bisimilarity [2], even if the two notions actually coincide, together with contextual equivalence. 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 is of the form $(\mathcal{E}, M, N)$ or $\mathcal{E}$, and 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 5. *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 the translation of environmental bisimilarity to first-order, 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 5. 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; and Γ_i is the i-th element in Γ.

For an 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 $(\mathcal{E}_2, N)$. Similarly, an environment $\mathcal{E}$ is split into $\mathcal{E}_1$ and $\mathcal{E}_2$. 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:

$$\frac{M \longmapsto_{\mathsf{n}} M'}{(\Gamma, M) \xrightarrow{\tau} (\Gamma, M')} \qquad\qquad \frac{\Gamma_i(C[\Gamma]) \longmapsto_{\mathsf{n}} M'}{\Gamma \xrightarrow{i,C} (\Gamma, M')} \tag{2}$$

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 5: 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.

Theorem 4. *$M \approx^{\mathsf{env}}_{\mathcal{E}} N$ iff $(\mathcal{E}_1, M) \approx (\mathcal{E}_2, N)$ and $\mathcal{E} \in \approx^{\mathsf{env}}$ iff $\mathcal{E}_1 \approx \mathcal{E}_2$.*

(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. A useful such technique is 'up to environment', which allows us to replace an environment with a larger one; $\mathsf{w}(\mathcal{R})$ is the smallest relation that includes $\mathcal{R}$ and such that, whenever $(V, \Gamma, M)\ \mathsf{w}(\mathcal{R})\ (W, \Delta, N)$ then also $(\Gamma, M)\ \mathsf{w}(\mathcal{R})\ (\Delta, N)$, where V and W are any values. (Here w stands for 'weakening' as, from Lemmas 2 and 4, if $(V, \Gamma, M) \approx (W, \Delta, N)$ then $(\Gamma, M) \approx (\Delta, N)$.)

Lemma 4. *Function w is compatible.*

Somehow 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: $(\Gamma, C_v[\Gamma], M)\ \mathsf{str}(\mathcal{R})\ (\Delta, C_v[\Delta], N)$ whenever $(\Gamma, M)\ \mathcal{R}\ (\Delta, N)$ and C_v is a value context (i.e., the outermost operator is an abstraction). We derive the compatibility up to of str in Theorem 5.

For up-to context, we need to distinguish between arbitrary contexts and evaluation contexts. There are indeed substitutivity 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 whose first hole is in redex position.

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

Theorem 5. *The set $\{\mathsf{str}, \mathcal{C}, \mathcal{C}_{\mathsf{e}}\}$ is $\mathbf{sp}$-compatible up to the identity function, and $\mathbf{wp}$-compatible up to $\mathbf{wp} \cup \mathbf{e}$ where $\mathbf{e} \triangleq (\mathcal{R} \mapsto \gtrsim\mathcal{R}\lesssim)$ is 'up to expansion'.*

For the proof, we establish the progression property separately for each function in $\{\mathsf{str}, \mathcal{C}, \mathcal{C}_{\mathsf{e}}\}$, using simple diagram-chasing arguments (together with induction on the structure of a context). Once more, the compatibility of the up to context functions entails also the substitutivity properties of environmental bisimilarity. In [22] the two aspects (substitutivity 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 function allows us to split the proof into two simpler proofs.

Remark 2. A transition system ensuring full abstraction as in Theorem 4 does not guarantee the compatibility of the up-to techniques specific to the language

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

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

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

$$\frac{(s;M) \longmapsto_{\mathsf{R}} (s';M')}{(s;E[M]) \longmapsto_{\mathsf{R}} (s';E[M'])}$$

Fig. 1. The imperative λ-calculus

in consideration. For instance, a simpler and maybe more natural alternative to the second transition in (2) is the following one:

$$\frac{}{\Gamma \xrightarrow{i,C} (\Gamma, \Gamma_i(C[\Gamma]))} \tag{3}$$

With this rule, full abstraction holds, but up-to context is unsound: for any Γ and Δ, the singleton relation $\{(\Gamma,\Delta)\}$ is a bisimulation up to $\mathcal{C}$: indeed, using rule (3), 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 (2) 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 [10,22]. The syntax of terms, values, and evaluation contexts, as well as the reduction semantics are given in Figure 1. 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 s,t 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 rewritten using $\mathsf{set}_\ell V$, or read using $\mathsf{get}_\ell V$ (the former instruction returns a value, namely the identity $I \triangleq \lambda x.x$, and the argument of the latter one is ignored). We denote the reflexive and transitive closure of $\longmapsto_{\mathsf{R}}$ by $\Longmapsto_{\mathsf{R}}$.

Note that in contrast with the languages in [10,22], 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 $(\mathsf{get}_\ell, \mathsf{set}_\ell)$.

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 $(s;M)\Longmapsto_{\mathsf{R}}\downarrow$. For the definition of contextual equivalence, we distinguish the cases of values and of arbitrary terms, because they have different substitutivity properties: values can be tested in arbitrary contexts, while arbitrary terms must be tested only in evaluation contexts. As in [22], we consider contexts that do not contain free locations (they can contain bound locations). We refer to [22] for more details on these aspects.

Definition 6. – *For values V, W, we write $(s;V)\equiv(t;W)$ when $(s;C[V])\Downarrow$ iff $(t;C[W])\Downarrow$, for all location-free context C.*
– *For terms M and N, we write $(s;M)\equiv(t;N)$ when $(s;E[M])\Downarrow$ iff $(t;E[N])\Downarrow$, for all location-free evaluation context 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 (2) 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 (2). 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., 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 set_ℓ and get_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 $\mathsf{cod}(r)$ the tuple of the contexts D_ℓ in r, we add $\mathsf{cod}(r)$ 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 set_ℓ and get_ℓ operators, and these allow manipulation of 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. Concerning additional up-to functions, specific to ΛR, 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 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

$$((s \uplus r[\Gamma']; \Gamma', M),\ (t \uplus r[\Delta']; \Delta', N))$$

such that $(s; \Gamma, M)\ \mathcal{R}\ (t; \Delta, N)$, and with $\Gamma' = \Gamma, \mathsf{getset}(r)$ and $\Delta' = \Delta, \mathsf{getset}(r)$.

Lemma 5. *The set* $\{\mathsf{w}, \mathsf{str}, \mathcal{C}_{\mathsf{e}}, \mathsf{store}, \mathcal{C}\}$ *is* $\mathbf{sp}$*-compatible up to the identity function and is* $\mathbf{wp}$*-compatible up to* $\mathbf{wp} \cup \mathbf{e}$.

The techniques $\mathcal{C}$ and $\mathcal{C}_{\mathsf{e}}$ allow substitutivity under location-free contexts, from which we can derive the soundness part of Theorem 6.

Theorem 6. $(s; M) \equiv (t; N)$ *iff* $(s; M) \approx (t; N)$.

Proof (sketch). Soundness ($\Leftarrow$) follows from congruence by $\mathcal{C}_{\mathsf{e}}$ (Lemmas 5 and 2) and completeness ($\Rightarrow$) is obtained by standard means. See [15] for details.

Note that substitutivity of bisimilarity is restricted either to values ($\mathcal{C}$), or to evaluation contexts ($\mathcal{C}_{\mathsf{e}}$). The following lemma provides a sufficient condition for a given law between arbitrary terms to be preserved by arbitrary contexts.

Lemma 6. *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 stores* s. *Then also* $(s; \Gamma, C[L]) \asymp (s; \Gamma, C[R])$, *for any store* s, *environment* Γ *and context* C.

Proof (sketch). We first prove a simplified result in which C is an evaluation context, using techniques $\mathcal{C}_{\mathsf{e}}$ and store. We then exploit this partial result together with up-to expansion to derive the general result. See [15] for more details.

We use this lemma 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 [10]. 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:

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

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.

This example is also considered in [22] where it is however modified to fit the up-to techniques considered in that paper. The latter are less powerful than those available here thanks to the theory of up-to techniques for first-order LTSs (e.g., up to expansion is not considered in [22]—its addition to environmental bisimulations is non-trivial, having stores and environments as parameters).

We consider two proofs of the example. In comparison with the proof in [22]: (i) we handle the original example from [10], 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 [22]. 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 6 to reach a variant similar to that of [22]: we make a 'thunk' out of the test in M, and we make N look similar. More precisely, let $\mathsf{test}_\ell \triangleq \lambda z.\texttt{if}\ !\ell\ \texttt{mod}\ 2 = 0\ \texttt{then}\ I\ \texttt{else}\ \Omega$, 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:

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

The initial pair of terms is generalised by adding any number of private locations, since M' can use itself to create more of them. Relation $\mathcal{R}$ is a weak bisimulation up to store, $\mathcal{C}$ and expansion. More details can be found in [15].

Second proof. Here we also preprocess the terms using Lemma 6, 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 just 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 $\mathcal{R}$ is a strong bisimulation *up to unfolding, store, weakening, strengthening, transitivity and context* (a technique unsound in the weak case):

$$\mathcal{S} \triangleq \{(M', N'')\} \cup \{(\ell \mapsto 2n, \mathsf{incr}_\ell, \mathsf{test}_\ell), (\emptyset, \mathsf{incr}_0, \mathsf{test}_0) \text{ s.t. } n \in \mathbb{N}\}$$

This relation uses a single location; there is one pair for each integer that can be stored in the 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. (Again, we refer to [15] for more details.)

The difference between the relation $\mathcal{R}$ in the first proof and the proofs in [10,22] is that $\mathcal{R}$ only requires locations that appear free in the tested terms; in contrast, the relations in [10,22] 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. The reason why, both in [10,22] 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.

Acknowledgement. The authors acknowledge support from the ANR projects 2010-BLAN-0305 PiCoq and 12IS02001 PACE.

References

1. Abadi, M., Gordon, A.D.: A bisimulation method for cryptographic protocols. In: Hankin, C. (ed.) ESOP 1998. LNCS, vol. 1381, pp. 12–26. Springer, Heidelberg (1998)
2. Abramsky, S.: The lazy lambda calculus. In: Turner, D. (ed.) Research Topics in Functional Programming, pp. 65–116. Addison-Wesley (1989)
3. Arun-Kumar, S., Hennessy, M.: An efficiency preorder for processes. Acta Informatica 29, 737–760 (1992)
4. Chaudhuri, K., Cimini, M., Miller, D.: Formalization of the bisimulation-up-to technique and its meta theory. Draft (2014)
5. Hirschkoff, D.: A full formalisation of pi-calculus theory in the calculus of constructions. In: Gunter, E.L., Felty, A.P. (eds.) TPHOLs 1997. LNCS, vol. 1275, pp. 153–169. Springer, Heidelberg (1997)
6. Hur, C.-K., Neis, G., Dreyer, D., Vafeiadis, V.: The power of parameterization in coinductive proof. In: POPL, pp. 193–206. ACM (2013)

7. Jeffrey, A., Rathke, J.: Towards a theory of bisimulation for local names. In: LICS, pp. 56–66 (1999)
8. Koutavas, V., Hennessy, M.: First-order reasoning for higher-order concurrency. Computer Languages, Systems & Structures 38(3), 242–277 (2012)
9. Koutavas, V., Levy, P.B., Sumii, E.: From applicative to environmental bisimulation. Electr. Notes Theor. Comput. Sci. 276, 215–235 (2011)
10. Koutavas, V., Wand, M.: Small bisimulations for reasoning about higher-order imperative programs. In: POPL 2006, pp. 141–152. ACM (2006)
11. Lassen, S.B.: Relational reasoning about contexts. In: Higher-order Operational Techniques in Semantics, pp. 91–135. Cambridge University Press (1998)
12. Lassen, S.B.: Relational Reasoning about Functions and Nondeterminism. PhD thesis, Department of Computer Science, University of Aarhus (1998)
13. Lassen, S.B.: Bisimulation in untyped lambda calculus: Böhm trees and bisimulation up to context. Electr. Notes Theor. Comput. Sci. 20, 346–374 (1999)
14. Lenisa, M.: Themes in Final Semantics. Ph.D. thesis, Universitá di Pisa (1998)
15. Madiot, J.-M., Pous, D., Sangiorgi, D.: Web appendix to this paper, `http://hal.inria.fr/hal-00990859`
16. Merro, M., Nardelli, F.Z.: Behavioral theory for mobile ambients. J. ACM 52(6), 961–1023 (2005)
17. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
18. Pohjola, J.Å., Parrow, J.: Bisimulation up-to techniques for psi-calculi. Draft (2014)
19. Pous, D., Sangiorgi, D.: Enhancements of the bisimulation proof method. In: Advanced Topics in Bisimulation and Coinduction. Cambridge University Press (2012)
20. Rot, J., Bonsangue, M., Rutten, J.: Coalgebraic bisimulation-up-to. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 369–381. Springer, Heidelberg (2013)
21. Sangiorgi, D.: On the bisimulation proof method. J. of MSCS 8, 447–479 (1998)
22. Sangiorgi, D., Kobayashi, N., Sumii, E.: Environmental bisimulations for higher-order languages. ACM Trans. Program. Lang. Syst. 33(1), 5 (2011)
23. Sangiorgi, D., Walker, D.: The Pi-Calculus: a theory of mobile processes. Cambridge University Press (2001)
24. Sumii, E., Pierce, B.C.: A bisimulation for dynamic sealing. Theor. Comput. Sci. 375(1-3), 169–192 (2007)
25. Sumii, E., Pierce, B.C.: A bisimulation for type abstraction and recursion. J. ACM 54(5) (2007)
26. Turner, N.D.: The polymorphic pi-calculus: Theory and Implementation. PhD thesis, Department of Computer Science, University of Edinburgh (1996)