Complete Lattices and Up-to Techniques

Damien Pous

Complete Lattices and Up-to Techniques*

Extended version (with proofs)**

Damien Pous

LIP: UMR CNRS - ENS Lyon - UCB Lyon - INRIA 5668, France

Abstract. We propose a theory of up-to techniques for proofs by coinduction, in the setting of complete lattices. This theory improves over existing results by providing a way to compose arbitrarily complex techniques with standard techniques, expressed using a very simple and modular semi-commutation property.
Complete lattices are enriched with monoid operations, so that we can recover standard results about labelled transitions systems and their associated behavioural equivalences at an abstract, "point-free" level.
Our theory gives for free a powerful method for validating up-to techniques. We use it to revisit up to contexts techniques, which are known to be difficult in the weak case: we show that it is sufficient to check basic conditions about each operator of the language, and then rely on an iteration technique to deduce general results for all contexts.

Introduction

Coinductive definitions are frequently used in order to define operational or contextual equivalences, in settings ranging from process algebra [12] to functional programming [10,11,19]. This approach relies on Knaster-Tarski's fixpoint theorem [22]: "*in a complete lattice, any order-preserving function has a greatest fixpoint, which is the least upper bound of the set of its post-fixpoints*". Hence, by defining an object x as the greatest fixpoint of an order-preserving function, we have a powerful technique to show that some object y is dominated by x: prove that y is dominated by some post-fixpoint. However, in some cases, the least post-fixpoint dominating y can be a "large" object: when reasoning about bisimilarity on a labelled transition system (LTS), the smallest bisimulation relating two processes has to contain all their reducts. Hence, checking that this relation is actually a bisimulation often turns out to be tedious. The aim of up-to techniques, as defined in [12,17], is to alleviate this task by defining functions f over relations such that any bisimulation "up to f" is contained in a bisimulation and hence in bisimilarity. These techniques have been widely used [11,21,6,13,19], and turn out to be essential in some cases.

* This work has been supported by the french project "ModyFiable", funded by ANR ARASSIA.

** This research report (LIP: RR2007-30) extends an abstract that has been published in Proc. APLAS 2007 [15].

In this paper, we generalise the theory of [17] to the abstract setting of complete lattices [3]. This allows us to ignore the technicalities of LTSs and binary relations, and to obtain a homogeneous theory where we only manipulate objects and order-preserving functions (called maps). The key notion is that of *compatible* maps, i.e., maps satisfying a very simple semi-commutation property. These maps, which correspond to up-to techniques, generalise the "respectful" functions of [17]. They enjoy the same nice compositional properties: we can construct sophisticated techniques from simpler ones. On the other hand, there are cases where compatible maps are not sufficient: we prove in [13] the correctness of a distributed abstract machine, where mechanisms introduced by an optimisation cannot be taken into account by standard techniques relying on compatible maps (e.g., *up to expansion* [1,20]); we have to resort to recent, and more sophisticated techniques [16] relying on termination hypotheses.

The powerful techniques of [16] cannot be expressed by means of compatible maps, which makes it difficult to combine them with other techniques: we have to establish again correctness of each combination. Our first contribution addresses this problem: we give a simple condition ensuring that the composition of an arbitrarily complex correct technique and a compatible map remains correct. While this result is not especially difficult, it greatly enhances both [17], where only compatible maps are considered, and [16], where the lack of compositionality renders the results quite ad-hoc, and their proofs unnecessarily complicated. We illustrate the benefits of this new approach in Sect. 4, by establishing an uncluttered generalisation of one of the main results from [16], and showing how to easily enrich the corresponding up-to technique with standard techniques.

We then refine our framework, by adding monoidal operations to complete lattices, together with a symmetry operator. In doing so, we obtain an abstract, point-free presentation of binary relations, which is well-suited to proofs by diagram chasing arguments. In this setting, an LTS is a collection $(\xrightarrow{\alpha})_{\alpha\in\mathcal{L}}$ of objects, indexed by some *labels*, and strong similarity is the largest object x such that the semi-commutation diagram (S) below is satisfied:

$$\begin{array}{ccc} \cdot & & x \\ {\scriptstyle\alpha}\downarrow & \sqsubseteq & \downarrow{\scriptstyle\alpha} \\ x & & \cdot \end{array} \qquad\qquad \begin{array}{ccc} \cdot & & x \\ {\scriptstyle\alpha}\downarrow & \sqsubseteq & \downarrow{\scriptstyle\alpha} \\ f(x) & & \cdot \end{array}$$

(S) (S_f)

There is an implicit universal quantification on all labels α, so that this diagram should be read $(S):\ \forall\alpha\in\mathcal{L},\ \xleftarrow{\alpha}\cdot x \sqsubseteq x\cdot\xleftarrow{\alpha}$ (where $(\cdot)$ is the law of the monoid, $\sqsubseteq$ is the partial order of the complete lattice, and $\xleftarrow{\alpha}$ denotes the converse of relation $\xrightarrow{\alpha}$). The second diagram, (S_f), illustrates the use of a map f as an up-to technique: "x satisfies (S) up to f". Intuitively, if $x \sqsubseteq f(x)$, it will be easier to check (S_f) than (S); the correctness of f should then ensure that x is dominated by some object satisfying (S).

By defining two other notions of diagrams, and using symmetry arguments, we show how to recover in a uniform way the standard behavioural preorders and equivalences (strong and weak bisimilarity, expansion [1]), together with

their associated up-to techniques. Notably, we can reduce the analysis of up-to techniques for those two-sided games to the study of their one-sided constituents.

Another advantage of working in this point-free setting is that it encompasses various cases, where objects are not necessarily simple binary relations. This includes *typed bisimulations* [21,8], where processes are related at a given type and/or in a given typing environment; and *environment bisimulations* [19], where environments are used to keep track of the observer's knowledge. Therefore, we obtain standard up-to techniques for these complicated settings, and more importantly, this gives a clear theory to guarantee correctness of up-to techniques that can be specific to these settings.

We then observe that maps over a complete lattice are an instance of complete lattice equipped with monoidal operations satisfying our requirements. We show that compatible maps, which are defined via a semi-commutation property, can be seen as the post-fixpoints of a *functor* (a map over maps). Therefore, our theory provides us for free with up-to techniques for compatible maps. We illustrate the use of such "second-order" techniques by considering up to context techniques; which are well-known for CCS or the π-calculus [21], and quite hard for functional languages [10,11,19]. Even in the simple case of CCS, (polyadic) contexts have a complex behaviour which renders them difficult to analyse. We show how to use an "up to iteration" technique in order to reduce the analysis of arbitrary contexts to that of the constructions of the language only. While we consider here the case of CCS, the resulting methodology is quite generic, and should be applicable to various other calculi (notably the π-calculus).

Outline. The abstract theory is developed in Sect. 1; we apply it to LTSs and behavioural preorders in Sect. 2. Section 3 is devoted to up to context techniques for CCS; we show in Sect. 4 how to combine a complex technique with compatible maps. We conclude with directions for future work in Sect. 5.

1 Maps and Fixpoints in Complete Lattices

1.1 Preliminary Definitions

We assume a *complete lattice*, that is, a tuple $\langle X, \sqsubseteq, \bigvee \rangle$, where $\sqsubseteq$ is a partial order over a set X (a reflexive, transitive and anti-symmetric relation), such that any subset Y of X has a *least upper bound* (*lub* for short) that we denote by $\bigvee Y$.

$$\forall y \in Y,\ y \sqsubseteq \bigvee Y \ , \qquad \forall x \in X,\ (\forall y \in Y,\ y \sqsubseteq x) \Rightarrow \bigvee Y \sqsubseteq x \ .$$

A function $f : X \to X$ is *order-preserving* if $\forall x, y \in X\ x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$; it is *continuous* if $\forall Y \subseteq X,\ Y \neq \emptyset \Rightarrow f(\bigvee Y) = \bigvee f(Y)$. We extend $\sqsubseteq$ and $\bigvee$ pointwise to functions: $f \sqsubseteq g$ if $\forall x \in X,\ f(x) \sqsubseteq g(x)$, and $\bigvee F : x \mapsto \bigvee \{f(x) \mid f \in F\}$ for any family F of functions. In the sequel, we only consider order-preserving functions, which we shall simply call *maps*. For any element

y and maps f, g, we define the following maps: $\mathrm{id}_X : x \mapsto x$; $\widehat{y} : x \mapsto y$; $f \circ g : x \mapsto f(g(x))$ and $f^\omega \triangleq \bigvee \{f^n \mid n \in \mathbb{N}\}$, where $f^0 \triangleq \mathrm{id}_X$ and $f^{n+1} \triangleq f \circ f^n$. We say that a map f is *extensive* if $\mathrm{id}_X \sqsubseteq f$.

We fix in the sequel a map s.

Definition 1.1. An *s-simulation* is an element x such that $x \sqsubseteq s(x)$. We denote by X_s the set of all s-simulations, *s-similarity* ($\boldsymbol{\nu} s$) is the lub of this set:

$$X_s \triangleq \{x \in X \mid x \sqsubseteq s(x)\} \ , \qquad \boldsymbol{\nu} s \triangleq \bigvee X_s \ .$$

Theorem 1.2 (Knaster-Tarski [22])**.** *$\boldsymbol{\nu} s$ is the greatest fixpoint of s: $\boldsymbol{\nu} s = s(\boldsymbol{\nu} s)$.*

Proof. We show that both $\boldsymbol{\nu} s$ and $s(\boldsymbol{\nu} s)$ are a post-fixpoints:

$$\begin{aligned}
&\forall x \in X_s,\ x \sqsubseteq \boldsymbol{\nu} s && \text{(by definition of } \boldsymbol{\nu} s) \\
&\forall x \in X_s,\ x \sqsubseteq s(x) \sqsubseteq s(\boldsymbol{\nu} s) && (s \text{ is order-preserving}) \\
&\boldsymbol{\nu} s \sqsubseteq s(\boldsymbol{\nu} s) && \text{(lubs property)} \\
&s(\boldsymbol{\nu} s) \sqsubseteq s(s(\boldsymbol{\nu} s)) && (s \text{ is order-preserving}) \\
&s(\boldsymbol{\nu} s) \in X_s && \text{(by definition of } X_s) \\
&s(\boldsymbol{\nu} s) \sqsubseteq \boldsymbol{\nu} s \ . && \text{(by definition of } \boldsymbol{\nu} s)
\end{aligned}$$

■

1.2 Up-to Techniques for Proofs by Coinduction

The previous definition gives the powerful *coinduction* proof method: in order to prove that $y \sqsubseteq \boldsymbol{\nu} s$, it suffices to find some y' such that $y \sqsubseteq y' \sqsubseteq s(y')$. The idea of up-to techniques is to replace s with a map s', such that:

- $s \sqsubseteq s'$ so that there are more s'-simulations than s-simulations; and
- $\boldsymbol{\nu} s' \sqsubseteq \boldsymbol{\nu} s$ so that the proof method remains correct.

At first, we restrict ourselves to maps of the form $s \circ f$ and focus on the map f.

Definition 1.3. A map f is *s-correct* if $\boldsymbol{\nu}\,(s \circ f) \sqsubseteq \boldsymbol{\nu} s$.
A map f is *s-correct via f'* if f' is an extensive map and $f'(X_{s \circ f}) \subseteq X_s$.
A map f is *s-compatible* if $f \circ s \sqsubseteq s \circ f$.

Proposition 1.4. (i) *Any s-compatible map f is s-correct via f^ω.*
(ii) *Any map is s-correct iff it is s-correct via some map.*

Proof. Let f be a compatible map; f^ω is extensive by definition. Let then $x \in X_{s \circ f} : x \sqsubseteq s(f(x))$. We prove $\forall n,\ f^n(x) \sqsubseteq s(f^{n+1}(x))$ by induction, using the compatibility of f. We conclude with properties of lubs: $f^\omega(x) \sqsubseteq s(f^\omega(x))$.

Any s-correct map f is correct via $f' = \mathrm{id}_X \vee \widehat{\boldsymbol{\nu} s}$: if $x \sqsubseteq s(f(x))$ then $x \sqsubseteq \boldsymbol{\nu} s$ since f is s-correct, and $f'(x) = \boldsymbol{\nu} s \in X_s$. Conversely, let f be an s-correct map via another map f'. We have $\boldsymbol{\nu}\,(s \circ f) \in X_{s \circ f}$, so that $f'(\boldsymbol{\nu}\,(s \circ f)) \in X_s$ and $f'(\boldsymbol{\nu}\,(s \circ f)) \sqsubseteq \boldsymbol{\nu} s$. Since f' is extensive, we have $\boldsymbol{\nu}\,(s \circ f) \sqsubseteq \boldsymbol{\nu} s$. ■

Intuitively, a map is correct via f' if its correctness can be proved using f' as a "witness function" – these witnesses will be required to establish Prop. 1.14 and Thm. 1.17 below. For example, in the case of an s-compatible map f, if $x \sqsubseteq s(f(x))$ then $f^\omega(x)$, which is an s-simulation, is the witness.

Remark 1.5. For any s-compatible map f, $f(\boldsymbol{\nu} s) \sqsubseteq \boldsymbol{\nu} s$. Hence s-compatible maps necessarily correspond to closure properties satisfied by $\boldsymbol{\nu} s$. This is not a sufficient condition: there are maps satisfying $f(\boldsymbol{\nu} s) \sqsubseteq \boldsymbol{\nu} s$ that are not s-correct.

There also exist s-correct maps that do not preserve $\boldsymbol{\nu} s$; we can however prove that for any map f' such that there exists an extensive s-correct map via f', $f'(\boldsymbol{\nu} s) = \boldsymbol{\nu} s$.

Proposition 1.6. *The family of s-compatible maps is stable under composition and lubs. It contains the identity, and constant maps $\widehat{x}$ with $x \in X_s$.*

These nice compositional properties are the main motivation behind compatible maps. They do not hold for correct maps (more generally, the map $t = \bigvee \{t \mid \boldsymbol{\nu} t \sqsubseteq \boldsymbol{\nu} s\}$ does not necessarily satisfy $\boldsymbol{\nu} t \sqsubseteq \boldsymbol{\nu} s$). On the other hand, correct maps allow more expressiveness: we can use any mathematical argument in order to prove the correctness of a map; we will for example use well-founded inductions in Sect. 4.

At this point, we have generalised to a rather abstract level the theory developed in [17] (this claim is justified by Sect. 1.4). Thm 1.8, which is our first improvement against [17], allows one to compose correct and compatible maps:

Lemma 1.7. *Let f, g be two maps.*
If f is s-compatible and g-compatible, then f is $(s \circ g)$-compatible.

Proof. We have $s \circ g \circ f \sqsubseteq s \circ f \circ g \sqsubseteq f \circ s \circ g$. ■

Theorem 1.8. *Let f be an s-compatible map, and g an s-correct map via g'.*
If f is g-compatible, then $(g \circ f)$ is s-correct via $(g' \circ f^\omega)$.

Proof. By Lemma 1.7, f is $(s \circ g)$-compatible, so that $f^\omega(X_{s \circ g \circ f}) \subseteq X_{s \circ g}$, by Prop. 1.4. Since $g'(X_{s \circ g}) \subseteq X_s$, we can conclude: $g'(f^\omega(X_{s \circ g \circ f})) \subseteq X_s$. ■

Notice that this theorem also holds without specifying the correction witnesses. The following lemma will be used in Sect. 2.2.

Lemma 1.9. *Let f, f', g' be three maps. If f is s-correct via f', g' is extensive, and g' preserves s-simulations ($g'(X_s) \subseteq X_s$), then f is s-correct via $(g' \circ f')$.*

As will be illustrated in Sect. 4, this important result allows one to focus on the heart of a complex technique, so that its proof remains tractable; and then to improve this technique with more standard techniques.

1.3 Conjunctions, Symmetry, and Internal Monoid

We now add some structure to complete lattices: conjunctions, which are already supported, symmetry, and monoidal laws.

Conjunctions. A complete lattice has both lubs and glbs: we denote by Y^l the set of lower bounds of a subset Y of X. The *greatest lower bound of* Y (*glb* for short), is the lub of this set:

$$Y^l \triangleq \{x \in X \mid \forall y \in Y,\ x \sqsubseteq y\} \ , \qquad \bigwedge Y \triangleq \bigvee Y^l \ .$$

We extend this definition pointwise to maps.

Lemma 1.10. *For any $x \in X$ and $Y \subseteq X$, $x \sqsubseteq \bigwedge Y$ iff $x \in Y^l$.*

Lemma 1.11. *For any family $\mathcal{Y}$ of subsets of X, $\bigvee \bigcap \mathcal{Y} \sqsubseteq \bigwedge \{\bigvee Y \mid Y \in \mathcal{Y}\}$.*

Proof. For any $Y \in \mathcal{Y}$, we have $\bigcap \mathcal{Y} \subseteq Y$ and hence, $\bigvee \bigcap \mathcal{Y} \sqsubseteq \bigvee Y$. Therefore, by Lemma 1.10, $\bigvee \bigcap \mathcal{Y} \sqsubseteq \bigwedge \{\bigvee Y \mid Y \in \mathcal{Y}\}$. ■

We fix in the sequel a set S of maps and focus on proof techniques for $\bigwedge S$. As will be illustrated in Sect. 2.2, this kind of maps corresponds to coinductive definitions based on a conjunction of several properties.

Lemma 1.12. *We have $X_{\bigwedge S} = \bigcap \{X_s \mid s \in S\}$ and $\boldsymbol{\nu} \bigwedge S \sqsubseteq \bigwedge \{\boldsymbol{\nu} s \mid s \in S\}$.*

Proof. Lemma 1.10 gives

$$\begin{aligned} x \in X_{\bigwedge S} \Leftrightarrow x \sqsubseteq \bigwedge S(x) &\Leftrightarrow \forall s \in S,\ x \sqsubseteq s(x) \\ &\Leftrightarrow \forall s \in S,\ x \in X_s \Leftrightarrow x \in \bigcap \{X_s \mid s \in S\} \ . \end{aligned}$$

Then, with Lemma 1.11, we have:

$$\boldsymbol{\nu} \bigwedge S = \bigvee X_{\bigwedge S} = \bigvee \bigcap \{X_s \mid s \in S\} \sqsubseteq \bigwedge \left\{ \bigvee X_s \mid s \in S \right\} = \bigwedge \{\boldsymbol{\nu} s \mid s \in S\} \ .$$

■

In general, $\boldsymbol{\nu} \bigwedge S \neq \bigwedge \{\boldsymbol{\nu} s \mid s \in S\}$; for example, in process algebras, 2-simulation and bisimulation do not coincide. Therefore, to obtain results about $\boldsymbol{\nu} \bigwedge S$, it is not sufficient to study the fixpoints $(\boldsymbol{\nu} s)_{s \in S}$ separately.

Proposition 1.13. *Any map that is s-compatible for all s in S is $\bigwedge S$-compatible.*

Notice that this is actually the dual of one point of Prop. 1.6: "the lub of a family of s-compatible maps is s-compatible".

Proof. For any such map f, we have

$$f \circ \bigwedge S \sqsubseteq \bigwedge \{f \circ s \mid s \in S\} \sqsubseteq \bigwedge \{s \circ f \mid s \in S\} = \bigwedge S \circ f \ .$$

■

Prop. 1.13 deals with compatible maps, and requires that the same map is used for all the components of S. We can relax these restrictions by working with correct maps, provided that they agree on a common witness: Again, this is related to the difference between 2-simulation and bisimulation:

Proposition 1.14. *Let $(f_s)_{s\in S}$ be a family of maps indexed by S and let f' be an extensive map; let $S_f \triangleq \{s \circ f_s \mid s \in S\}$.*

If f_s is s-correct via f' for all s of S, then $\boldsymbol{\nu}\bigwedge S_f \sqsubseteq \boldsymbol{\nu}\bigwedge S$.

Proof. We have $\boldsymbol{\nu}\bigwedge S_f = \bigwedge S_f(\boldsymbol{\nu}\bigwedge S_f)$ and then

$$\forall s \in S,\ \boldsymbol{\nu}\bigwedge S_f \sqsubseteq (s \circ f_s)\,(\boldsymbol{\nu}\bigwedge S_f) \qquad \text{(Lemma 1.10)}$$

$$\forall s \in S,\ f'(\boldsymbol{\nu}\bigwedge S_f) \in X_s \qquad (f_s \text{ is } s\text{-correct via } f')$$

$$f'(\boldsymbol{\nu}\bigwedge S_f) \in X_{\bigwedge S} \qquad \text{(Lemma 1.12)}$$

f' being extensive, we can conclude: $\boldsymbol{\nu}\bigwedge S_f \sqsubseteq f'(\boldsymbol{\nu}\bigwedge S_f) \sqsubseteq \boldsymbol{\nu}\bigwedge S$. ■

Although Prop. 1.14 does not define a $\bigwedge S$-correct map, it actually defines an up-to technique for $\bigwedge S$: a priori, $\bigwedge S \sqsubseteq \bigwedge S_f$, so that $\bigwedge S_f$-simulations are easier to construct than $\bigwedge S$-simulations.

Symmetry. Let $\bar{\cdot}$ be an order-preserving involution ($\forall x,\ \overline{\overline{x}} = x$). For any map f, we define

$$\overline{Y} \triangleq \{\overline{y} \mid y \in Y\}\ , \qquad \overline{f} \triangleq \bar{\cdot} \circ f \circ \bar{\cdot} : x \mapsto \overline{f(\overline{x})}\ , \qquad \overleftrightarrow{f} \triangleq f \wedge \overline{f}\ .$$

We call $\overline{x}$ the *converse of* x and we say that an element x (resp. a map f) is *symmetric* if $x = \overline{x}$ (resp. $f = \overline{f}$).

Lemma 1.15. *Let $x, y \in X$, $f, g : X \hookrightarrow X$ and $Y \subseteq X$. We have:*

(i) $\overline{\overline{f}} = f$;
(ii) $\overline{f}(x) = \overline{f}(\overline{x})$; $\overline{f \circ g} = \overline{f} \circ \overline{g}$; $\overline{\mathrm{id}_X} = \mathrm{id}_X$; $\widehat{\overline{x}} = \widehat{\overline{x}}$.
(iii) $x \sqsubseteq y$ *iff* $\overline{x} \sqsubseteq \overline{y}$; $f \sqsubseteq g$ *iff* $\overline{f} \sqsubseteq \overline{g}$; $\overline{\bigvee Y} = \bigvee \overline{Y}$.

Using the previous properties, we can relate up-to techniques for s and $\overline{s}$:

Proposition 1.16. *We have $\overline{X_s} = X_{\overline{s}}$, $\overline{\boldsymbol{\nu} s} = \boldsymbol{\nu}\overline{s}$ and for any maps f, f',*

(i) *f is s-correct (via f') if and only if $\overline{f}$ is $\overline{s}$-correct (via $\overline{f'}$),*
(ii) *f is s-compatible if and only if $\overline{f}$ is $\overline{s}$-compatible.*

Proof. We have $x \in X_s \Leftrightarrow x \sqsubseteq s(x) \Leftrightarrow \overline{x} \sqsubseteq \overline{s(x)} \Leftrightarrow \overline{x} \sqsubseteq \overline{s}(\overline{x}) \Leftrightarrow \overline{x} \in X_{\overline{s}}$. Then, $\overline{\boldsymbol{\nu} s} \triangleq \overline{\bigvee X_s} = \bigvee \overline{X_s} = \bigvee X_{\overline{s}} \triangleq \boldsymbol{\nu}\overline{s}$. By Lemma 1.15(i), it suffices to show the direct implication in last two points

(i) If $\boldsymbol{\nu}\,(s \circ f) \sqsubseteq \boldsymbol{\nu} s$ then $\boldsymbol{\nu}\left(\overline{s} \circ \overline{f}\right) = \boldsymbol{\nu}\left(\overline{s \circ f}\right) = \overline{\boldsymbol{\nu}\,(s \circ f)} \sqsubseteq \overline{\boldsymbol{\nu} s} = \boldsymbol{\nu}\overline{s}$.
If $f'(X_{s\circ f}) \subseteq X_s$ then $\overline{f'}(X_{\overline{s}\circ\overline{f}}) = \overline{f'}(\overline{X_{s\circ f}}) = \overline{f'(X_{s\circ f})} \subseteq \overline{X_s} = X_{\overline{s}}$.
(ii) If $f \circ s \sqsubseteq s \circ f$ then $\overline{f} \circ \overline{s} = \overline{f \circ s} \sqsubseteq \overline{s \circ f} = \overline{s} \circ \overline{f}$. ■

We can finally combine these properties with Prop. 1.14 and reduce the problem of finding up-to techniques for $\overleftrightarrow{s}$ to that of finding up-to techniques for s. We illustrate this in Sect. 2.2, by deriving up-to techniques for weak bisimulation from techniques for weak simulation. An immediate corollary of Prop. 1.13 is that any map symmetric s-compatible map is $\overleftrightarrow{s}$-compatible. This result extends to correct maps as follows:

Theorem 1.17. *For any s-correct map f via a symmetric map f', we have*

$$f'\left(X_{\overleftrightarrow{s \circ f}}\right) \subseteq X_{\overleftrightarrow{s}} \ , \quad \text{and} \qquad \boldsymbol{\nu}\overleftrightarrow{s \circ f} \sqsubseteq \boldsymbol{\nu}^{\overleftrightarrow{s}} \ .$$

Proof. By using Prop. 1.16, we check that $\{f, \overline{f}\}$ and $f' = \overline{f'}$ satisfy the hypotheses of Prop. 1.14, for $S = \{s, \overline{s}\}$. ■

Corollary 1.18. *Let f be an s-correct map via a symmetric map.*
If x is symmetric, and $x \sqsubseteq s(f(x))$, then $x \sqsubseteq \boldsymbol{\nu}^{\overleftrightarrow{s}}$.

Internal monoid. Suppose that the complete lattice $\langle X, \sqsubseteq, \bigvee \rangle$ is actually a *monoidal complete lattice*, i.e., that X is equipped with an associative product $(\cdot)$ with neutral element e, such that:

$$\forall x, y, x', y' \in X, \ x \sqsubseteq x' \wedge y \sqsubseteq y' \Rightarrow x \cdot y \sqsubseteq x' \cdot y' \ . \tag{1}$$

The *iteration* (resp. *strict iteration*) of an element x is defined by $x^\star \triangleq \bigvee_{n \in \mathbb{N}} x^n$ (resp. $x^+ \triangleq \bigvee_{n>0} x^n$), where $x^0 \triangleq e$ and $x^{n+1} \triangleq x \cdot x^n$. Iterations and product are extended pointwise to maps: $f \mathbin{\hat{\cdot}} g : x \mapsto f(x) \cdot g(x)$, and $f^\star : x \mapsto f(x)^\star$.

Definition 1.19. An element x is *reflexive* if $e \sqsubseteq x$; it is *transitive* if $x \cdot x \sqsubseteq x$. We say that s *preserves the monoid* $\langle X, \cdot, e \rangle$ if e is an s-simulation and

$$\forall x, y \in X, \ s(x) \cdot s(y) \sqsubseteq s(x \cdot y) \ . \tag{2}$$

Proposition 1.20. *If s preserves the monoid, then:*

(i) *the product of two s-simulations is an s-simulation;*
(ii) *s-similarity $(\boldsymbol{\nu} s)$ is reflexive and transitive;*
(iii) *for any s-compatible maps f, g, $f \mathbin{\hat{\cdot}} g$ and $f^\star$ are s-compatible.*

Proof. (i) If $x \sqsubseteq s(x)$ and $y \sqsubseteq s(y)$, then $x \cdot y \sqsubseteq s(x) \cdot s(y) \sqsubseteq s(x \cdot y)$ by (1) and (2).
(ii) By the previous point, $\boldsymbol{\nu} s \cdot \boldsymbol{\nu} s$ is an s-simulation, so that $\boldsymbol{\nu} s \cdot \boldsymbol{\nu} s \sqsubseteq \boldsymbol{\nu} s$. Moreover, $e \sqsubseteq \boldsymbol{\nu} s$ by hypothesis.
(iii) If $f \circ s \sqsubseteq s \circ f$ and $g \circ s \sqsubseteq s \circ g$, then we have

$$(f \mathbin{\hat{\cdot}} g) \circ s = (f \circ s) \mathbin{\hat{\cdot}} (g \circ s) \sqsubseteq (s \circ f) \mathbin{\hat{\cdot}} (s \circ g) \sqsubseteq s \circ (f \mathbin{\hat{\cdot}} g)$$

by (1) and (2), so that $f \mathbin{\hat{\cdot}} g$ is s-compatible. The fact that $f^\star$ is s-compatible follows from Prop. 1.6. ■

1.4 Progressions

We now reformulate the theory of [17] in the abstract setting of complete lattices, and we relate it to our theory.

Definition 1.21. A *progression* is a binary relation $\rightarrowtail$ over X, such that:

$$\forall x, y, z \in X,\ x \rightarrowtail y \text{ and } y \sqsubseteq z \text{ entail } x \rightarrowtail z\ , \tag{3}$$

$$\forall Y \subseteq X,\ \forall z \in X,\ \text{if } \forall y \in Y,\ y \rightarrowtail z,\ \text{then } \bigvee Y \rightarrowtail z\ . \tag{4}$$

A progression is *closed* if it moreover satisfies:

$$\forall x, y, z \in X,\ x \sqsubseteq y \text{ and } y \rightarrowtail z \text{ entail } x \rightarrowtail z\ . \tag{5}$$

We fix in the sequel a progression $\rightarrowtail$.

Definition 1.22. An element $x \in X$ is a $\rightarrowtail$*-simulation* if $x \rightarrowtail x$.
The $\rightarrowtail$*-similarity* is the lub of all $\rightarrowtail$*-simulations*: $\boldsymbol{\nu}_{\rightarrowtail} \triangleq \bigvee_{x \rightarrowtail x} x$.

Proposition 1.23. $\rightarrowtail$*-similarity is the greatest* $\rightarrowtail$*-simulation.*

Definition 1.24. Let f be a map.

- f *is* $\rightarrowtail$*-correct* if $\forall x \in X,\ x \rightarrowtail f(x)$ entails $x \sqsubseteq \boldsymbol{\nu}_{\rightarrowtail}$;
- f *is* $\rightarrowtail$*-respectful* if $\forall x, y \in X,\ x \rightarrowtail y$ entails $f(x) \rightarrowtail f(y)$.
- $\rightarrowtail$ *preserves the monoid* $\langle X, \cdot, e\rangle$ if $e \rightarrowtail e$ and

$$\forall x, x', y, y' \in X,\ x \rightarrowtail x',\ y \rightarrowtail y' \Rightarrow x \cdot y \rightarrowtail x' \cdot y'\ . \tag{2'}$$

Notice that our notion of *respectful* map slightly differs from that found in [17,21]: the two notions coincide when the progression we consider is contained in the partial order $(x \rightarrowtail y \Rightarrow x \sqsubseteq y)$.

Lemma 1.25. *Let f be a $\rightarrowtail$-respectful map and x an element of X.*
If $x \rightarrowtail f(x)$, then $f^{\omega}(x)$ is a $\rightarrowtail$-simulation.

The following result correspond to [17, Thm. 2.11]:

Theorem 1.26. *Any $\rightarrowtail$-respectful map is $\rightarrowtail$-correct.*

The result below correspond to Thm. 1.8, which has no equivalent in [17].

Theorem 1.27. *Let f be a $\rightarrowtail$-respectful map, and let g be a $\rightarrowtail$-correct map.*
If f is g-compatible, then $(g \circ f)$ is $\rightarrowtail$-correct.

Proof sketch. We define $\rightarrowtail_g \triangleq \{\langle x, y\rangle \mid x \rightarrowtail g(y)\}$, which is a progression since g is order-preserving. We prove that f is $\rightarrowtail_g$-respectful:

$$x \rightarrowtail_g y \Leftrightarrow x \rightarrowtail g(y) \Rightarrow f(x) \rightarrowtail f(g(y)) \sqsubseteq g(f(y)) \Leftrightarrow f(x) \rightarrowtail_g f(y)\ .$$

Hence, f is $\rightarrowtail_g$-correct by Thm. 1.26, and the correction of g gives $\boldsymbol{\nu}_{\rightarrowtail_g} \sqsubseteq \boldsymbol{\nu}_{\rightarrowtail}$ so that we can conclude. ■

From maps to progressions

Definition 1.28. We call *progression associated to* s the following relation:

$$\rightarrowtail_s \triangleq \{\langle x, y\rangle \mid x \sqsubseteq s(y)\} \ .$$

Proposition 1.29. (i) $\rightarrowtail_s$ *is a closed progression;*
(ii) *For all* $x, y \in X$, $x \rightarrowtail_s y$ *iff* $x \sqsubseteq s(y)$.
(iii) *the* $\rightarrowtail_s$*-simulations are the* s*-simulations;* $\boldsymbol{\nu}_{\rightarrowtail} = \boldsymbol{\nu} s$;
(iv) *the* $\rightarrowtail_s$*-respectful maps are the* s*-compatible maps;*
(v) *the* $\rightarrowtail_s$*-correct maps are the* s*-correct maps;*
(vi) $\rightarrowtail_s$ *preserves the monoid iff* s *preserves the monoid.*

Proof. (i), (ii) and (iii) are straightforward.
(iv) – let f be a $\rightarrowtail_s$-respectful map. Let $x \in X$, we have:

$$\begin{aligned} s(x) &\sqsubseteq s(x) && \text{(reflexivity)} \\ s(x) &\rightarrowtail_s x && \text{(by definition)} \\ f(s(x)) &\rightarrowtail_s f(x) && (f \text{ respectful}) \\ f(s(x)) &\sqsubseteq s(f(x)) \ . && \text{(by definition)} \end{aligned}$$

Hence, $f \circ s \sqsubseteq s \circ f$.
– Conversely, suppose that f is s-compatible, and $x \rightarrowtail_s y$:

$$\begin{aligned} x &\sqsubseteq s(y) && \text{(by definition)} \\ f(x) &\sqsubseteq f(s(y)) && (f \text{ is order-preserving}) \\ f(x) &\sqsubseteq s(f(y)) && \text{(compatibility of } f) \\ f(x) &\rightarrowtail_s f(y) \ . && \text{(by definition)} \end{aligned}$$

(v) By (ii), we have $x \rightarrowtail f(x)$ iff $x \sqsubseteq s(f(x))$.
(vi) – Suppose that $\rightarrowtail_s$ preserves the monoid, and let $x, y \in X$. We have $s(x) \rightarrowtail_s x$ and $s(y) \rightarrowtail_s y$, so that $s(x) \cdot s(y) \rightarrowtail_s x \cdot y$, i.e., $s(x) \cdot s(y) \sqsubseteq s(x \cdot y)$.
– Conversely, if s preserves the monoid, $x \rightarrowtail_s x'$ and $y \rightarrowtail_s y'$, then $x \cdot y \sqsubseteq s(x') \cdot s(y')$ by (1), and $x \cdot y \sqsubseteq s(x' \cdot y')$ by transitivity, i.e., $x \cdot y \rightarrowtail_s x' \cdot y'$. ■

From progressions to maps

Definition 1.30. For all $x \in X$, we denote by $\lfloor x \rfloor$ the set $\{y \in X \mid y \rightarrowtail x\}$. We call *map associated to* $\rightarrowtail$ the map $s_{\rightarrowtail} : y \mapsto \bigvee \lfloor x \rfloor$.

Notice that if we apply this construction to $\rightarrowtail_s$, we recover s: $s = s_{\rightarrowtail_s}$.

Lemma 1.31. *The map* $s_{\rightarrowtail}$ *is order-preserving; for all* $x \in X$, $\bigvee \lfloor x \rfloor \rightarrowtail x$.

Lemma 1.32. *We have* $\rightarrowtail = \rightarrowtail_{s_{\rightarrowtail}}$ *if and only if* $\rightarrowtail$ *is closed.*

Proof. ($\Leftarrow$) By Proposition 1.29(i) $\rightarrowtail_{s_{\rightarrowtail}}$ is always closed.

($\Rightarrow$) If $x \rightarrowtail y$ then $x \in \lfloor y \rfloor$, whence $x \sqsubseteq \bigvee \lfloor y \rfloor$, i.e., $x \rightarrowtail_{s_{\rightarrowtail}} y$. Conversely, if $x \rightarrowtail_{s_{\rightarrowtail}} y$ then $x \sqsubseteq \bigvee \lfloor y \rfloor \rightarrowtail y$, whence $x \rightarrowtail y$ since $\rightarrowtail$ is closed. ■

Proposition 1.33. *If $\rightarrowtail$ is closed, then:*

(i) *the $\rightarrowtail$-simulations are the $s_{\rightarrowtail}$-simulations; $\boldsymbol{\nu} s_{\rightarrowtail} = \boldsymbol{\nu}_{\rightarrowtail}$;*
(ii) *the $\rightarrowtail$-respectful maps are the $s_{\rightarrowtail}$-compatible maps;*
(iii) *the $\rightarrowtail$-correct maps are the $s_{\rightarrowtail}$-correct maps.*
(iv) *$\rightarrowtail$ preserves the monoid iff $s_{\rightarrowtail}$ preserves the monoid.*

Proof. By Prop. 1.32, $\rightarrowtail \; = \; \rightarrowtail_{s_{\rightarrowtail}}$. Therefore, it is sufficient to go through Prop. 1.29. For example, for the first point, the $s_{\rightarrowtail}$-simulations are the $\rightarrowtail_{s_{\rightarrowtail}}$-simulations, which are the $\rightarrowtail$-simulations. ■

1.5 Up-to Techniques for Compatible Maps

Denoting by $X^{\langle X \rangle}$ the set of (order-preserving) maps over X, $\langle X^{\langle X \rangle}, \sqsubseteq, \bigvee, \circ, \mathrm{id}_X \rangle$ forms a monoidal complete lattice. Therefore, we can apply the previous theory in order to capture certain properties of maps. In particular, that of being s-compatible: for any map s, define the following relation over maps:

$$f \overset{s}{\rightsquigarrow} f' \quad \text{if} \quad f \circ s \sqsubseteq s \circ f' \ .$$

Lemma 1.34. *$\overset{s}{\rightsquigarrow}$ is a progression relation.*

Since s is order-preserving, $\overset{s}{\rightsquigarrow}$ is a progression relation, whose simulations are exactly the s-compatible maps. Moreover, when s comes from a progression ($s = s_{\rightarrowtail}$), we have $f \overset{s_{\rightarrowtail}}{\rightsquigarrow} f'$ iff $\forall x, y \in X,\ x \rightarrowtail y \Rightarrow f(x) \rightarrowtail f'(y)$.

Lemma 1.35. *For any map s, $\overset{s}{\rightsquigarrow}$ preserves the monoid $\langle X^{\langle X \rangle}, \circ, \mathrm{id}_X \rangle$.*

Proof. If $f \overset{s}{\rightsquigarrow} f'$ and $g \overset{s}{\rightsquigarrow} g'$, then $f \circ g \circ s \sqsubseteq f \circ s \circ g' \sqsubseteq s \circ f' \circ g'$, i.e., $f \circ g \overset{s}{\rightsquigarrow} f' \circ g'$; moreover, we clearly have $\mathrm{id}_X \overset{s}{\rightsquigarrow} \mathrm{id}_X$. ■

Theorem 1.36. *Let f, g be two maps.*

(i) *If the product $(\cdot)$ preserves s and $f \overset{s}{\rightsquigarrow} f^\star$, then $f^\star$ is s-compatible.*
(ii) *If $f \overset{s}{\rightsquigarrow} f^\omega$ and f is continuous, then f^ω is s-compatible.*
(iii) *If $f \overset{s}{\rightsquigarrow} g \circ f^\omega$, where g is s-compatible, extensive and idempotent ($g \circ g = g$), and f is g-compatible, then $g \circ f^\omega$ is s-compatible.*

Proof. Call *functor* any (order-preserving) map φ over maps; we say that a functor is *respectful* when it is compatible w.r.t. $\overset{s}{\rightsquigarrow}$. Recall that $\widehat{g}$ is the constant functor to g, and that $(\hat{\circ})$ is the pointwise extension of $(\circ)$ to functors.

(i) By Lemma 1.35 and Prop. 1.20, $\varphi = \widehat{\mathrm{id}_X^\star} \mathbin{\hat{\circ}} \mathrm{id}_{X^{\langle X \rangle}} : f \mapsto f^\star$ is respectful, being the product of two respectful functors:
   – the constant functor to $\mathrm{id}_X^\star$, this map being s-compatible by Prop. 1.20;

– and the identity functor $\mathrm{id}_{X^{\langle X\rangle}}$, which is always respectful.
Therefore, f is "s-compatible up to the respectful functor φ", so that $\varphi^\omega(f)$ is s-compatible, by Prop. 1.4. We finally check that $\varphi^\omega(f) = f^\star$.

(ii) By Lemma 1.35 and Prop. 1.20, the functor $\omega \triangleq \mathrm{id}^\star_{X^{\langle X\rangle}} : f \mapsto f^\omega$ is respectful (iteration (*) is done w.r.t ($\circ$)). By Prop. 1.4, $\omega^\omega(f)$ is s-compatible, and we check that $\omega^\omega(f) = f^\omega$, f being continuous.

(iii) Using similar arguments, $\varphi = \widehat{g} \mathbin{\hat{\circ}} \omega : f \mapsto g \circ f^\omega$ is respectful, and $\varphi^\omega(f)$ is s-compatible. We then have to check that $\varphi^\omega(f) = \varphi(f) = g \circ f^\omega$: f being g-compatible, we have $f^\omega \circ g \sqsubseteq g \circ f^\omega$; which entails $(g \circ f^\omega)^\omega = g \circ f^\omega$ since g is extensive and idempotent. Therefore, we have $\varphi^2(f) = g \circ (g \circ f^\omega)^\omega = g \circ f^\omega = \varphi(f)$, which leads to $\varphi^\omega(f) = \varphi(f)$. ■

The first point generalises [21, Lemma 2.3.16]; we illustrate the use of (ii) and (iii) in Sect. 3. In (iii), the main hypotheses are the progression property and s-compatibility of g: other hypotheses are only used in order to simplify computations, so that the actual s-compatible map we obtain is $g \circ f^\omega$.

2 Bisimilarity in Monoidal Lattices with Symmetry

We assume a *continuous monoidal complete lattice with symmetry*, that is, a monoidal complete lattice $\langle X, \sqsubseteq, \bigvee, \cdot, e\rangle$, whose product distributes over arbitrary lubs: $(\forall Y, Z \sqsubseteq X,\ \bigvee Y \cdot \bigvee Z = \bigvee \{y \cdot z \mid y \in Y,\ z \in Z\})$, equipped with a map $\bar{\cdot}$ such that $\forall x,\ \overline{\overline{x}} = x$ and $\forall x, y,\ \overline{x \cdot y} = \overline{y} \cdot \overline{x}$.

Although we denote by $x, y \ldots$ the elements of X, they should really be thought of as "abstract relations" so that we shall call them *relations* in the sequel (we employ letters R, S for "set-theoretic relations" of Sect. 3 and 4).

We let α range over the elements of a fixed set $\mathcal{L}$ of *labels*, and we assume a *labelled transition system (LTS)*, that is, a collection $(\xrightarrow{\alpha})_{\alpha\in\mathcal{L}}$ of relations indexed by $\mathcal{L}$. Intuitively, $\xrightarrow{\alpha}$ represents the set of transitions along label α. Among the elements of $\mathcal{L}$, we distinguish the *silent action*, denoted by τ; we let a range over the elements of $\mathcal{L}^{\mathrm{v}} \triangleq \mathcal{L} \setminus \{\tau\}$, called *visible labels*. For $\alpha \in \mathcal{L}$ we define the following *weak transition relations*:

$$\xrightarrow{\widehat{\alpha}} \triangleq \begin{cases} \xrightarrow{\tau} \vee\, e & \text{if } \alpha = \tau \ , \\ \xrightarrow{\alpha} & \text{otherwise ;} \end{cases} \qquad \stackrel{\alpha}{\Rightarrow} \triangleq \xrightarrow{\tau}{}^\star \cdot \xrightarrow{\alpha} \cdot \xrightarrow{\tau}{}^\star \ ; \quad \stackrel{\widehat{\alpha}}{\Rightarrow} \triangleq \xrightarrow{\tau}{}^\star \cdot \xrightarrow{\widehat{\alpha}} \cdot \xrightarrow{\tau}{}^\star \ .$$

Notice the following properties: $\stackrel{\widehat{\tau}}{\Rightarrow} = \xrightarrow{\tau}{}^\star$, $\stackrel{\tau}{\Rightarrow} = \xrightarrow{\tau}{}^+$, $\stackrel{\widehat{a}}{\Rightarrow} = \stackrel{a}{\Rightarrow}$. The converses of such relations will be denoted by the corresponding reversed arrows.

2.1 One-Sided Behavioural Preorders

In order to define behavioural preorders, we construct four maps in Fig. 1, based on four different progressions. Their meaning can be recovered by considering the simulations they define: **s** yields strong simulation games, where actions are exactly matched (diagram (S) in the introduction); **e** yields games corresponding

$$
\begin{array}{lll}
\mathbf{s}: & x \rightarrowtail_{\mathbf{s}} y & \text{if } x \sqsubseteq y \text{ and } \forall \alpha \in \mathcal{L},\ \overset{\alpha}{\leftarrow} \cdot x \sqsubseteq y \cdot \overset{\alpha}{\leftarrow}\ ; \\
\mathbf{e}: & x \rightarrowtail_{\mathbf{e}} y & \text{if } x \sqsubseteq y \text{ and } \forall \alpha \in \mathcal{L},\ \overset{\alpha}{\leftarrow} \cdot x \sqsubseteq y \cdot \overset{\widehat{\alpha}}{\leftarrow}\ ; \\
\mathbf{w}: & x \rightarrowtail_{\mathbf{w}} y & \text{if } x \sqsubseteq y \text{ and } \forall \alpha \in \mathcal{L},\ \overset{\alpha}{\leftarrow} \cdot x \sqsubseteq y \cdot \overset{\widehat{\alpha}}{\Leftarrow}\ ; \\
\mathbf{w}_{\mathrm{t}}: & x \rightarrowtail_{\mathbf{w}_{\mathrm{t}}} y & \text{if } x \sqsubseteq y,\ \overset{\tau}{\leftarrow} \cdot x \sqsubseteq y \cdot \overset{\widehat{\tau}}{\Leftarrow}\ , \text{ and } \forall a \in \mathcal{L}^{\mathrm{v}},\ \overset{a}{\leftarrow} \cdot x \sqsubseteq y^{\star} \cdot \overset{\widehat{a}}{\Leftarrow}\ .
\end{array}
$$

Fig. 1. Maps and progressions for left-to-right simulation-like games.

to the left-to-right part of an *expansion* [1,20] game, where it is allowed not to move on silent challenges; and $\mathbf{w}$ yields weak simulations games, where one can answer "modulo silent transitions". The map ($\mathbf{w}_{\mathrm{t}}$) is a variant of $\mathbf{w}$, which allows one to answer up to transitivity on visible challenges. We have $\mathbf{s} \sqsubseteq \mathbf{e} \sqsubseteq \mathbf{w} \sqsubseteq \mathbf{w}_{\mathrm{t}}$, so that $X_{\mathbf{s}} \subseteq X_{\mathbf{e}} \subseteq X_{\mathbf{w}} \subseteq X_{\mathbf{w}_{\mathrm{t}}}$, and $\boldsymbol{\nu}\mathbf{s} \subseteq \boldsymbol{\nu}\mathbf{e} \subseteq \boldsymbol{\nu}\mathbf{w} \subseteq \boldsymbol{\nu}\mathbf{w}_{\mathrm{t}}$.

In these definitions, the requirement $x \sqsubseteq y$ may seem worthless; it is however required in almost all compatibility results we shall prove below; the absence of this requirement is notably the source of unnecessary complications in [17].

Lemma 2.1. *Relations from Fig. 1 are progressions relations.*

Proof. Straightforward, distribution of the product over arbitrary lubs is required for the second clause of the defintion of a progression, however. ■

Lemma 2.2. *Maps* $\mathbf{s}$ *and* $\mathbf{e}$ *preserve the monoid.*

Proof. Straightforward diagram chasing arguments. ■

The following proposition collects standard up-to techniques that can be used with these maps. Maps $\mathbf{s}$ and $\mathbf{e}$ preserve the monoid, so that they enjoy the properties stated in Prop. 1.20: the corresponding greatest fixpoints are reflexive and transitive, and they support the powerful "up to transitivity" technique (i). This is not the case for $\mathbf{w}$: if it was preserving the monoid, the "weak up to weak" technique would be correct, which is not true [20]. We can however show directly that $\mathbf{w}$-simulations are closed under composition ($\cdot$), and that they support "up to expansion" on the left, and "up to weak" on the right (ii). Map $\mathbf{w}_{\mathrm{t}}$ is actually an up-to technique for $\mathbf{w}$: the similarities associated to those maps coincide (iii). Intuitively, transitivity can be allowed on visible actions, since these are played in a one-to-one correspondence.

Proposition 2.3. (i) *The reflexive transitive map* $\mathrm{id}_X^{\star}$ *is* $\mathbf{s}$*- and* $\mathbf{e}$*-compatible.*
(ii) *For any* $x_e \in X_{\mathbf{e}}$ *and* $x_w \in X_{\mathbf{w}}$*, the map* $y \mapsto x_e \cdot y \cdot x_w$ *is* $\mathbf{w}$*-compatible; this map is* $\mathbf{w}_t$*-compatible whenever* x_e *and* x_w *are reflexive.*
(iii) *For any* $\mathbf{w}_t$*-simulation* x*,* $x^{\star}$ *is a* $\mathbf{w}$*-simulation;* $\boldsymbol{\nu}\mathbf{w}_t = \boldsymbol{\nu}\mathbf{w}$*.*

Proof. (i) From Prop. 1.20, and Lemma 2.2.
(ii) We first show that any $\mathbf{w}$-simulation x satisfies $\forall \alpha \in \mathcal{L},\ \overset{\widehat{\alpha}}{\Leftarrow} \cdot x \sqsubseteq x \cdot \overset{\widehat{\alpha}}{\Leftarrow}$, the result follows easily.

(iii) By two inductions on n, we prove that $\forall n \in \mathbb{N},\ \overset{\hat{i}}{\Leftarrow} \cdot x^n \sqsubseteq x^n \cdot \overset{\hat{i}}{\Leftarrow}$ and $\forall n \in \mathbb{N},\ \overset{a}{\Leftarrow} \cdot x^n \sqsubseteq x^\star \cdot \overset{a}{\Leftarrow}$. ■

2.2 Handling Two-Sided Games

To study "reversed games" we just use the converses of the previous maps; for example, the map $\overline{\mathbf{w}}$ defines the same games as $\mathbf{w}$, from right to left: x is a $\overline{\mathbf{w}}$-simulation iff $\overline{x} \rightarrowtail_{\mathbf{w}} \overline{x}$. Using the results of Sect. 1.3 we can then combine left-to-right maps with right-to-left maps and obtain standard two-sided games:

$$\sim \triangleq \boldsymbol{\nu}^{\overleftrightarrow{\mathbf{s}}} \qquad \asymp \triangleq \boldsymbol{\nu}^{\overleftrightarrow{\mathbf{e}}} \qquad \gtrsim \triangleq \boldsymbol{\nu}(\mathbf{e} \wedge \overline{\mathbf{w}}) \qquad \approx \triangleq \boldsymbol{\nu}^{\overleftrightarrow{\mathbf{w}}}$$

Strong bisimilarity ($\sim$), *bi-expansion* ($\asymp$) [5] and *weak-bisimilarity* ($\approx$) are symmetric, reflexive and transitive; *expansion* [1,20] ($\gtrsim$) is reflexive and transitive; we have

$$\sim \sqsubseteq \asymp \sqsubseteq \gtrsim \sqsubseteq \approx \ .$$

Remark 2.4. Here we chose to follow the definition of bisimilarity where a bisimulation is a relation x such that x and $\overline{x}$ are simulations. Another standard definition consists in restricting the notion of bisimulations to symmetric simulations. We could mimic this definition by letting $\overleftrightarrow{s} \triangleq s \wedge i$ (we have that a relation x is an i-simulation ($x \sqsubseteq \overline{x}$) if and only if it is symmetric). Although this choice slightly restrict the set of compatible maps, we could adapt Thms. 2.6 and 4.5 to work with this definition.

Before transferring our techniques from one-sided to two-sided games, we introduce the notion of *closure*, that we use as an abstraction in order to cope with the up-to context techniques we shall define in Sect. 3. Notice that the continuity and extensivity hypotheses are not required for Thm. 2.6 to hold.

Definition 2.5. A *closure* is a continuous, extensive and symmetric map $\mathcal{C}$, such that $\forall x, y \in X,\ \mathcal{C}(x \cdot y) \sqsubseteq \mathcal{C}(x) \cdot \mathcal{C}(y)$.

We now can recover standard techniques for behavioural preorders or equivalences; we explain them below.

Theorem 2.6. *Let $\mathcal{C}$ be a closure.*

(i) *If $\mathcal{C}$ is $\mathbf{s}$-compatible, $x \mapsto (\mathcal{C}(x) \vee \sim)^\star$ is $\overleftrightarrow{\mathbf{s}}$-compatible*
(ii) *If $\mathcal{C}$ is $\mathbf{e}$-compatible, $x \mapsto (\mathcal{C}(x) \vee \asymp)^\star$ is $\overleftrightarrow{\mathbf{e}}$-compatible*
(iii) *If $\mathcal{C}$ is $\mathbf{e}$- and $\mathbf{w}$-compatible, then* $\boldsymbol{\nu}\left(\mathbf{e} \circ (\mathcal{C} \vee \widehat{\gtrsim})^\star \wedge \overline{\mathbf{w}} \circ (\widehat{\gtrsim} \mathbin{\hat{\cdot}} \mathcal{C} \mathbin{\hat{\cdot}} \widehat{\gtrsim})\right) = \gtrsim$.
(iv) *If $\mathcal{C}$ is $\mathbf{w}$-compatible, $x \mapsto \gtrsim \cdot \mathcal{C}(x) \cdot \precsim$ is $\overleftrightarrow{\mathbf{w}}$-compatible.*
(v) *If $\mathcal{C}$ is $\mathbf{w}$-compatible, $x \mapsto \gtrsim \cdot \mathcal{C}(x) \cdot \approx$ is $\mathbf{w}$-correct via a symmetric map.*
(vi) $\boldsymbol{\nu}^{\overleftrightarrow{\mathbf{w}_t}} = \approx$ *. If $\mathcal{C}$ is $\mathbf{w}_t$-compatible, $x \mapsto \gtrsim \cdot \mathcal{C}(x) \cdot \approx$ is $\mathbf{w}_t$-correct via a symmetric map.*

Proof. (i) By Props. 1.6 and 2.3, this map is $\mathbf{s}$-compatible; being symmetric, it is also $\overline{\mathbf{s}}$-compatible. We conclude with Prop. 1.13.

$\alpha \in \mathcal{L} \qquad a \in \mathcal{L}^{\mathrm{v}}$

$\overline{\tau} = \tau \qquad \overline{\overline{a}} = a$

$p ::= \mathbf{0} \mid \alpha.p \mid p|p \mid (\nu a)p \mid !p$

$$\frac{p \xrightarrow{\alpha} p'}{p \mid q \xrightarrow{\alpha} p' \mid q} \qquad \frac{q \xrightarrow{\alpha} q'}{p \mid q \xrightarrow{\alpha} p \mid q'} \qquad \frac{p \xrightarrow{a} p' \qquad q \xrightarrow{\overline{a}} q'}{p \mid q \xrightarrow{\tau} p' \mid q'}$$

$$\frac{}{\alpha.p \xrightarrow{\alpha} p} \qquad \frac{p \xrightarrow{\alpha} p'}{(\nu a)p \xrightarrow{\alpha} (\nu a)p'} \alpha \neq a, \overline{a} \qquad \frac{!p \mid p \xrightarrow{\alpha} p'}{!p \xrightarrow{\alpha} p'}$$

Fig. 2. Calculus of Communicating Systems (CCS)

(ii) Identical to the previous point.

(iii) Let $f_e = (\mathcal{C} \vee \widehat{\gtrsim})^\star$; f_e is **e**-compatible and hence, **e**-correct via f_e^ω. Let $f_w = \widehat{\eqsim} \mathbin{\hat{}} \mathcal{C} \mathbin{\hat{}} \widehat{\precsim}$; f_w is **w**-compatible, and hence, **w**-correct via f_w^ω. We then obtain that $\overline{f_w}$ is $\overline{\mathbf{w}}$-correct via $f' = (\mathrm{id}_X \vee \widehat{\gtrsim})^\star \circ \overline{f_w}^\omega$, by Lemma 1.9, We check that $f' = f_e^\omega$ and we apply Prop. 1.14.

(iv) This map is symmetric and **w**-compatible.

(v) Let $f : x \mapsto \gtrsim \cdot \mathcal{C}(x) \cdot \approx$; f is **w**-compatible, and hence **w**-correct via f^ω. by Lemma 1.9, it is also correct via the map $(\widehat{\approx} \mathbin{\hat{}} \mathrm{id}_X) \circ f^\omega = \widehat{\approx} \mathbin{\hat{}} \mathcal{C}^\omega \mathbin{\hat{}} \widehat{\approx}$, which is symmetric.

(vi) If x is a $\overleftrightarrow{\mathbf{w}}_{\mathrm{t}}$-simulation, then $x^\star$ is a $\overleftrightarrow{\mathbf{w}}$-simulation; the second point is similar to the previous one. ■

Intuitively, we may think of $\mathcal{C}(R)$ as being the closure of R under some set of contexts. (i) states that up-to transitivity and contexts is allowed for strong bisimilarity. This corresponds to the left diagram below: if x is symmetric and satisfies this diagram, then x is contained in $\sim$. The standard up to expansion and contexts for weak bisimulation is stated in (iv) and slightly improved in (v); notice that we need for that to use the notion of correct map: this map is not $\overleftrightarrow{\mathbf{w}}$-compatible. Technique (v) appears on the second diagram below. Finally, (vi) allows us to work up to transitivity on visible actions; which is depicted on the last two diagrams below

$$\begin{array}{ccc} \cdot & & x \\ {\scriptstyle\alpha}\downarrow & \sqsubseteq & \downarrow{\scriptstyle\alpha} \\ \cdot & & (\mathcal{C}(x) \vee \sim)^\star \cdot \end{array} \qquad \begin{array}{ccc} \cdot & & x \\ {\scriptstyle\alpha}\downarrow & \sqsubseteq & \Downarrow{\scriptstyle\widehat{\alpha}} \\ \gtrsim \cdot \mathcal{C}(x) \cdot \approx \cdot & & \end{array} \qquad \begin{array}{ccc} \cdot & & x \\ {\scriptstyle\tau}\downarrow & \sqsubseteq & \Downarrow{\scriptstyle\widehat{\tau}} \\ \gtrsim \cdot \mathcal{C}(x) \cdot \approx \cdot & & \end{array} \qquad \begin{array}{ccc} \cdot & & x \\ {\scriptstyle a}\downarrow & \sqsubseteq & \Downarrow{\scriptstyle a} \\ (\mathcal{C}(x) \vee \approx)^\star \cdot & & \end{array}$$

3 Congruence and Up to Context Techniques in CCS

We now look at "up to context" techniques, which provide an example of application of the results from Sect. 1.5. We need for that to instantiate the previous framework: contexts do not make sense in a point-free setting.

3.1 The case of sum-free CCS

We first restrict ourselves to the case of sum-free CCS [12], whose syntax and semantics are recalled in Fig. 2. The sum operator is the source of irregularities

in the weak case; we show how to handle this operator in Sect.3.2. Moreover, we chose replication (!) rather than recursive definitions in order to get an algebra which is closer the π-calculus.

We denote by $\mathcal{P}$ the set of processes, and we let R, S range over the set $\mathcal{R}$ of binary relations over $\mathcal{P}$. We write $p\ R\ q$ when $\langle p, q\rangle$ belongs to R. We denote by I the reflexive relation: $\{\langle p, p\rangle \mid p \in \mathcal{P}\}$. The composition of R and S is the relation $R \cdot S \triangleq \{\langle p, r\rangle \mid \exists q,\ p\ R\ q \text{ and } q\ S\ r\}$; the converse of R is $\overline{R} \triangleq \{\langle p, q\rangle \mid q\ R\ p\}$. We finally equip relations with set-theoretic inclusion ($\subseteq$) and union ($\bigcup$), so that $\langle \mathcal{R}, \subseteq, \bigcup, \cdot, I, \bar{\cdot}\rangle$ forms a monoidal complete lattice with symmetry.

For any natural number n, a *context with arity* n is a function $c : \mathcal{P}^n \to \mathcal{P}$, whose application to a n-uple of processes $p_1, \ldots, p_n$ is denoted by $c[p_1, \ldots, p_n]$. We associate to such context the following map (which is actually a closure):

$$\lfloor c \rfloor : R \mapsto \{\langle c[p_1, \ldots, p_n],\ c[q_1, \ldots, q_n]\rangle \mid \forall i \leq n,\ p_i\ R\ q_i\}$$

This notation is extended to sets C of contexts, by letting $\lfloor C \rfloor \triangleq \bigcup_{c \in C} \lfloor c \rfloor$.

Definition 3.1. We define the following *initial* contexts:

$$\mathbf{0} : p \mapsto \mathbf{0} \qquad | : p, q \mapsto p|q \qquad \alpha. : p \mapsto \alpha.p \qquad (\nu a) : p \mapsto (\nu a)p \qquad ! : p \mapsto !p$$

We gather these in the set $C_i \triangleq \{\mathrm{id}_{\mathcal{P}}, \mathbf{0}, |, !\} \cup \{\alpha. \mid \alpha \in \mathcal{L}\} \cup \{(\nu a) \mid a \in \mathcal{L}^{\mathrm{v}}\}$, and we call *closure under CCS contexts* the map $\mathcal{C}_{ccs} \triangleq \lfloor C_i \rfloor^{\omega}$.

Initial vs. Monadic Contexts. $\mathcal{C}_{ccs}(R)$ is actually the closure of R under arbitrary polyadic CCS contexts: we can show that $p\ \mathcal{C}_{ccs}(R)\ q$ iff p and q can be obtained by replacing some occurrences of $\mathbf{0}$ in a process with processes related by R. A different approach is adopted in [21]: the family C_m of *monadic CCS contexts* is defined; it consists in arbitrary CCS contexts, where the argument is used at most once. The map $\mathcal{C}_{ccs}$ can then be recovered by transitive closure: we have $\mathcal{C}_{ccs} \subseteq \lfloor C_m \rfloor^{\star}$. It has to be noticed that polyadic contexts cannot be avoided when we study the correctness of such maps: the monadic replication context (!) "evolves" by reduction into a polyadic context. In order to be able to consider only monadic contexts, a lemma corresponding to Thm 1.36(i) is used in [21], so that the proof in the strong case – reformulated into our setting – amounts to proving $\lfloor C_m \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \lfloor C_m \rfloor^{\star}$, i.e., $\forall c \in C_m$, $\lfloor c \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \lfloor C_m \rfloor^{\star}$, which is done by structural induction on context c (recall that $f \overset{\mathbf{s}}{\rightsquigarrow} f'$ iff $R \rightarrowtail_{\mathbf{s}} S$ entail $f(R) \rightarrowtail_{\mathbf{s}} f'(S)$). This approach does not scale to the weak case however, where up to transitivity is not correct, so that Thm 1.36(i) cannot no longer be used. Therefore, [21] suggests to work with polyadic contexts from the beginning, which is tedious and happens to require more attention than expected, as will be shown below.

Focusing on initial contexts makes it possible to reach $\mathcal{C}_{ccs}$ by iteration (Thm 1.36(ii)) rather than transitive closure, so that the extension to the weak case is not problematic. Moreover, initial contexts are much simpler than monadic contexts: the argument is almost at the top of the term, so that it is really easy to figure out the transitions of $c[p_1, \ldots, p_n]$. We give a detailed proof of the following theorem to illustrate the benefits of this approach.

Theorem 3.2. *The closure $\mathcal{C}_{ccs}$ is* **s***-compatible.*

Proof. By Thm.1.36(ii), it suffices to show $\lfloor C_i \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \mathcal{C}_{ccs}$, i.e., $\forall c \in C_i,\ \lfloor c \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \mathcal{C}_{ccs}$. We study each context of C_i separately, and we show

$$\lfloor \mathrm{id}_{\mathcal{P}} \rfloor = \mathrm{id}_{\mathcal{R}} \overset{\mathbf{s}}{\rightsquigarrow} \mathrm{id}_{\mathcal{R}} \qquad \lfloor \mathbf{0} \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \lfloor \mathbf{0} \rfloor \qquad \lfloor \alpha. \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \mathrm{id}_{\mathcal{R}}$$
$$\lfloor (\nu a) \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \lfloor (\nu a) \rfloor \qquad \lfloor | \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \lfloor | \rfloor \qquad \lfloor ! \rfloor \overset{\mathbf{s}}{\rightsquigarrow} \lfloor | \rfloor^{\omega} \circ (\lfloor ! \rfloor \cup \mathrm{id}_{\mathcal{R}})$$

(all maps used on the right of the above progression are contained in $\mathcal{C}_{ccs}$). Let R, S such that $R \rightarrowtail_{\mathbf{s}} S$, in each case, we suppose $u\ \lfloor c \rfloor(R)\ v$ and $u \overset{\alpha}{\to} u'$, and we have to find some v' such that $v \overset{\alpha}{\to} v'$ and $u'\ \lfloor c' \rfloor(S)\ v'$.

$\mathrm{id}_{\mathcal{R}}$, $\lfloor \mathbf{0} \rfloor$: straightforward.

$\lfloor \alpha. \rfloor$: $u = \alpha'.p \overset{\alpha}{\to} u'$, $v = \alpha'.q$ with $p\ R\ q$. Necessarily, $\alpha = \alpha'$ and $u' = p$. We hence have $v = \alpha.q \overset{\alpha}{\to} q$, with $p\ \mathrm{id}_{\mathcal{R}}(S)\ q$, (recall that $R \rightarrowtail_{\mathbf{s}} S$ entails $R \subseteq S$).

$\lfloor (\nu a) \rfloor$: $u = (\nu a)p \overset{\alpha}{\to} u'$, $v = (\nu a)q$ with $p\ R\ q$. Inferences rules impose $u' = (\nu a)p'$ where $p \overset{\alpha}{\to} p'$ and $\alpha \neq a, \overline{a}$. Since $p\ R\ q$, we obtain q' such that $q \overset{\alpha}{\to} q'$ and $p'\ S\ q'$, and we check that $v \overset{\alpha}{\to} v' = (\nu a)q'$, with $u'\ \lfloor (\nu a) \rfloor(S)\ v'$.

$\lfloor | \rfloor$: $u = p_1|p_2 \overset{\alpha}{\to} u'$, $v = q_1|q_2$ with $p_1\ R\ q_1$ and $p_2\ R\ q_2$. According to the inference rules in the case of a parallel composition, there are three cases:

- $u' = p_1'|p_2$ with $p_1 \overset{\alpha}{\to} p_1'$. Since $R \rightarrowtail_{\mathbf{s}} S$, $q_1 \overset{\alpha}{\to} q_1'$ with $p_1'\ S\ q_1'$. We check that $v \overset{\alpha}{\to} v' = q_1'|q_2$ and $u'\ \lfloor | \rfloor(S)\ v'$ (again we use $R \rightarrowtail_{\mathbf{s}} S \Rightarrow R \subseteq S$).
- $u' = p_1|p_2'$ with $p_2 \overset{\alpha}{\to} p_2'$, which is identical to the previous case.
- $u' = p_1'|p_2'$ with $p_1 \overset{a}{\to} p_1'$, $p_2 \overset{\overline{a}}{\to} p_2'$, and $\alpha = \tau$. We have $q_1 \overset{a}{\to} q_1'\ q_2 \overset{\overline{a}}{\to} q_2'$ with $p_1'\ S\ q_1'$ and $p_2'\ S\ q_2'$; so that $v \overset{\tau}{\to} v' = q_1'|q_2'$ and $u'\ \lfloor | \rfloor(S)\ v'$.

$\lfloor ! \rfloor$: $u = !p \overset{\alpha}{\to} u'$, $v = !q$ with $p\ R\ q$. By structural induction on inferences rules, there are two cases:

- $u' = !p|p^k|p'|p^{k'}$ with $p \overset{\alpha}{\to} p'$ (p^k denotes the parallel composition of k copies of p). We deduce $q \overset{\alpha}{\to} q'$ with $p'\ S\ q'$, and we check that $v \overset{\alpha}{\to} v' = !q|q^k|q'|q^{k'}$ and $u'\ \lfloor | \rfloor^{k+k'+1} \circ (\lfloor ! \rfloor \cup \mathrm{id}_{\mathcal{R}})(S)\ v'$.
- $u' = !p|p^k|p_0|p^{k'}|p_1|p^{k''}$ with $p \overset{a}{\to} p_0$, $p \overset{\overline{a}}{\to} p_1$ and $\alpha = \tau$. We deduce $q \overset{a}{\to} q_0$ and $q \overset{\overline{a}}{\to} q_1$ with $p_0\ S\ q_0$ and $p_1\ S\ q_1$. We check that $v \overset{\tau}{\to} v' = !q|q^k|q_0|q^{k'}|q_1|q^{k''}$, where $u'\ \lfloor | \rfloor^{k+k'+k''+1} \circ (\lfloor ! \rfloor \cup \mathrm{id}_{\mathcal{R}})(S)\ v'$. ∎

Contrarily to what is announced in [21, Lem. 2.4.52], $\mathcal{C}_{ccs}$ is not **w**-compatible: consider for example $R = \{\langle \tau.a, a \rangle\} \cup I$; although $R \rightarrowtail_{\mathbf{e}} R$, $\mathcal{C}_{ccs}(R) \rightarrowtail_{\mathbf{e}} \mathcal{C}_{ccs}(R)$ does not hold: the challenge $!\tau.a|a \overset{\tau}{\leftarrow} !\tau.a\ \lfloor ! \rfloor(R)\ !a$ cannot be answered in $\mathcal{C}_{ccs}(R)$ since $!a$ cannot move; we first have to rewrite $!a$ into $!a|a$. This is possible up to $\sim$: unfolding of replications is contained in strong similarity. [21] should thus be corrected by working modulo unfolding of replications, the corresponding proof would be *really* tedious however. In our setting, it suffices to use Thm. 1.36(iii): we work "up to iteration and a compatible map".

Theorem 3.3. *$R \mapsto \sim \cdot \mathcal{C}_{ccs}(R) \cdot \sim$ is an* **e***- and* **w***-compatible closure.*

$R = \{\langle a, (\nu b)(b.a|\bar{b})\rangle, \langle b.a, b\rangle, \langle (\nu b)(b|\bar{b}), \mathbf{0}\rangle, \langle (\nu b)\mathbf{0}, \mathbf{0}\rangle\}$
$c : p \mapsto (\nu b)(p|\bar{b})$

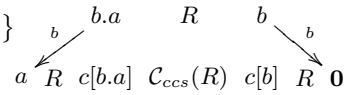


Fig. 3. Closure $\mathcal{C}_{ccs}$ is not $\mathbf{w}_\mathrm{t}$-correct.

*Proof (**w**-compatibility).* Take $g : r \mapsto \sim \cdot R \cdot \sim$; g is **w**-compatible, extensive and idempotent; moreover, $\mathcal{C}_{ccs}$ being **s**-compatible, $\mathcal{C}_{ccs}(\sim) \subseteq \sim$, and $\mathcal{C}_{ccs}$ is g-compatible. Hence, by Thm.1.36(iii), it suffices to show $\forall c \in C_i,\ \lfloor c \rfloor \overset{\mathbf{w}}{\rightsquigarrow} g \circ \mathcal{C}_{ccs}$.

Like previously, $\lfloor \mathbf{0} \rfloor \overset{\mathbf{w}}{\rightsquigarrow} \lfloor \mathbf{0} \rfloor$, $\lfloor | \rfloor \overset{\mathbf{w}}{\rightsquigarrow} \lfloor | \rfloor$, $\lfloor \alpha. \rfloor \overset{\mathbf{w}}{\rightsquigarrow} \mathrm{id}_{\mathcal{R}}$, and $\lfloor (\nu a) \rfloor \overset{\mathbf{w}}{\rightsquigarrow} \lfloor (\nu a) \rfloor$; we detail the case of the replication, for which we need the map g. Consider R, S such that $R \rightarrowtail_{\mathbf{w}} S$, we have to show $\lfloor ! \rfloor(R) \rightarrowtail_{\mathbf{w}} \sim \cdot \mathcal{C}_{ccs}(S) \cdot \sim$. Suppose that $p\ R\ q$ and $!p \xrightarrow{\alpha} p'$; there are two cases:

- $p' = !p|p^k|p_0|p^{k'}$ with $p \xrightarrow{\alpha} p_0$ (p^k denotes the parallel composition of k copies of p). Since $R \overset{\mathbf{w}}{\rightsquigarrow} S$, we deduce $q \overset{\widehat{\alpha}}{\Rightarrow} q_0$ with $p_0\ S\ q_0$. There are two cases:
  - $q \overset{\alpha}{\Rightarrow} q_0$, and we check that $!q \overset{\alpha}{\Rightarrow} q' = !q|q^k|q_0|q^{k'}$, where $p'\ \mathcal{C}_{ccs}(S)\ q'$.
  - $q = q_0$ (and $\alpha = \tau$), in that case, $!q$ cannot move, this is where we have reason modulo $\sim$: $!q \sim q' = !q|q^{k+1+k'}$, and $p'\ \mathcal{C}_{ccs}(S)\ q' \sim !q$.
- $p' = !p|p^k|p_0|p^{k'}|p_1|p^{k''}$ with $p \xrightarrow{a} p_0$ and $p \xrightarrow{\bar{a}} p_1$ ($\alpha = \tau$). Since $R \overset{\mathbf{w}}{\rightsquigarrow} S$, we deduce $q \overset{a}{\Rightarrow} q_0$ and $q \overset{\bar{a}}{\Rightarrow} q_1$ with $p_0\ S\ q_0$ and $p_1\ S\ q_1$. We check that $!q \overset{\tau}{\Rightarrow} q' = !q|q^k|q_0|q^{k'}|q_1|q^{k''}$, where $p'\ \mathcal{C}_{ccs}(S)\ q'$. ■

The proof of **e**-compatibility follows exactly the same lines; notice that the problematic situation, requiring to use g, cannot arise in the strong case.

A negative result. Rather surprisingly, $\mathcal{C}_{ccs}$ is not $\mathbf{w}_\mathrm{t}$-correct: a counterexample [18] is depicted on Fig. 3.1, where R is not contained in $\mathbf{w}_\mathrm{t}$-similarity while R is a $(\mathbf{w}_\mathrm{t} \circ \mathcal{C}_{ccs})$-simulation. The point is that $\lfloor | \rfloor \overset{\mathbf{w}_\mathrm{t}}{\rightsquigarrow} \mathcal{C}_{ccs}$ does not hold: since parallel composition is able to "transform" two visible actions into a silent action, up to transitivity is brought from visible challenges – where it is allowed by $\mathbf{w}_\mathrm{t}$, to silent challenges – where it is not

More precisely, we are stuck in the case of a communication, when we try to show that $R \rightarrowtail_{\mathbf{w}_\mathrm{t}} S$ entail $\lfloor | \rfloor(R) \rightarrowtail_{\mathbf{w}_\mathrm{t}} \mathcal{C}_{ccs}(S)$: using notations from the proof of Thm. 3.2, we have $u' = p'_1|p'_2$ with $p_1 \xrightarrow{a} p'_1$, $p_2 \xrightarrow{\bar{a}} p'_2$, and $\alpha = \tau$; however, the hypothesis $R \rightarrowtail_{\mathbf{w}_\mathrm{t}} S$ gives $q_1 \overset{a}{\Rightarrow} q'_1\ q_2 \overset{\bar{a}}{\Rightarrow} q'_2$ with $p'_1\ S^\star\ q'_1$ and $p'_2\ S^\star\ q'_2$ (rather than $p'_1\ S\ q'_1$ and $p'_2\ S\ q'_2$ with **w**); therefore we have $v \overset{\tau}{\Rightarrow} v' = q'_1|q'_2$ with $u'\ \lfloor | \rfloor(S^\star)\ v'$; being in a silent challenge, this is not sufficient: we have to prove that $u'\ \lfloor | \rfloor(S)\ v'$.

This shows that maps inducing the same fixpoint (recall that $\boldsymbol{\nu}\mathbf{w} = \boldsymbol{\nu}\mathbf{w}_\mathrm{t}$) may define different sets of compatible or correct maps. At a pragmatic level, this reveals the existence of a trade-off between the ability to use up to context and

up to transitivity. More importantly, it shows that from the point of view of up-to techniques, weak bisimilarity is different from "strong bisimilarity on the weak LTS ($\stackrel{\widehat{\alpha}}{\Rightarrow}$)": the relation R from Fig. 3.1 also satisfies $\forall \alpha,\ \stackrel{\widehat{\alpha}}{\Leftarrow} \cdot R \subseteq \mathcal{C}_{ccs}(R)^\star \cdot \stackrel{\widehat{\alpha}}{\Leftarrow}$.

3.2 Handling the sum operator.

We omitted the sum operator in order to obtain a rather uniform presentation. We show how to deal with this operator, which is known to be slightly problematic in the weak case, due to the preemptive power of silent action. We therefore extend our syntax of CCS with the sum $(p+q)$, and we add the following two rules to those from Fig. 2:

$$p ::= \ldots \mid p+p \qquad \frac{p \stackrel{\alpha}{\to} p'}{p+q \stackrel{\alpha}{\to} p'} \qquad \frac{q \stackrel{\alpha}{\to} q'}{p+q \stackrel{\alpha}{\to} q'}$$

The initial context corresponding to this construction is $+ : p, q \mapsto p+q$, so that we can define $C_i^+ \triangleq C_i \cup \{+\}$, and $\mathcal{C}_{ccs}^+ \triangleq \lfloor C_i^+ \rfloor^\omega$. For the strong case, we have $\lfloor + \rfloor \stackrel{\mathbf{s}}{\rightsquigarrow} \mathrm{id}_\mathcal{R}$, so that we immediately obtain the **s**-compatibility of $\mathcal{C}_{ccs}^+$.

This cannot be the case in the weak case: it is well-known that $\approx$ is not a congruence w.r.t. the sum operator. In our approach, the problem arise when we try to prove $\lfloor + \rfloor \stackrel{\mathbf{w}}{\rightsquigarrow} \mathcal{C}_{ccs}^+$: if $p_1 + p_2\ \lfloor + \rfloor(R)\ q_1 + q_2$, and $p_1 + q_1 \stackrel{\tau}{\to} p_1'$ with $p_1 \stackrel{\tau}{\to} p_1'$, then q_1 may not move: $q_1 = q_1'$ with $p_1'\ S\ q_1'$, and we cannot relate p_1' with a reduct of $q_1 + q_2$.

We actually have to use *non-degenerate* contexts [21], where all arguments are "protected" by a prefix. In order to obtain the corresponding closure, we introduce the following (initial) contexts, for any process r and prefixes $(\alpha_i)_{i \leq n}$:

$$r + \Sigma_i\ \alpha_i. : p_1, \ldots, p_n \mapsto r + \alpha_1.p_1 + \cdots + \alpha_n.p_n \ .$$

We can then define $C_i^{nd} \triangleq C_i \cup \{r + \Sigma_i\ \alpha_i. \mid \forall r, (\alpha_i)_i\}$, and $\mathcal{C}_{ccs}^{nd} \triangleq \lfloor C_i^{nd} \rfloor^\omega$. We check that $\lfloor r + \Sigma_i\ \alpha_i. \rfloor \stackrel{\mathbf{w}}{\rightsquigarrow} \mathrm{id}_\mathcal{R} \cup \widehat{I}$, so that $\mathcal{C}_{ccs}^{nd}$ is **w**-compatible (map $\widehat{I} : R \mapsto I$ is contained in $\mathcal{C}_{ccs}^{nd}$). The same argument leads to the **e**-compatibility of $\mathcal{C}_{ccs}^{nd}$.

Notice that an immediate consequence of the previous results is that $\sim$ is closed under all CCS contexts ($\mathcal{C}_{ccs}^+$), and that $\asymp$, $\gtrsim$ and $\approx$ are closed under all non degenerate CCS contexts ($\mathcal{C}_{ccs}^{nd}$).

4 Going Beyond Expansion: Termination Hypotheses.

In recent work [16], we proved that we can use up to transitivity and go beyond expansion – even on silent challenges – provided that some termination hypotheses are satisfied. In this section, we generalise the most important of these techniques (that has actually been used in [13]), and show how to integrate it with previously defined techniques. We say that a relation $\succ$ *terminates* if there exists no infinite sequence $(p_i)_{i \in \mathbb{N}}$ such that $\forall i \in \mathbb{N},\ p_i \succ p_{i+1}$.

We start by a technical lemma expressing the commutation property on which the technique relies.

Lemma 4.1. *Let $R, S, \to$ and $\hookrightarrow$ be four relations. If $S \subseteq R$, and $S^+ \cdot \to^+$ terminates, then*

$$\begin{cases} \leftarrow \cdot\, R \subseteq S^\star \cdot R \cdot \leftarrow^\star & \text{(H)} \\ \hookleftarrow \cdot\, R \subseteq R^\star \cdot \hookleftarrow \cdot \leftarrow^\star & \text{(H')} \end{cases} \textit{ entail } \quad \hookleftarrow \cdot \leftarrow^\star \cdot R^\star \subseteq R^\star \cdot \hookleftarrow \cdot \leftarrow^\star \ .$$

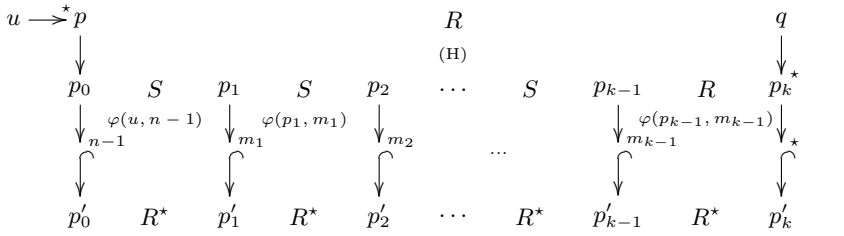
Proof. We actually prove $\hookleftarrow \cdot \leftarrow^\star \cdot R \subseteq R^\star \cdot \hookleftarrow \cdot \leftarrow^\star$, which leads to the desired result by a simple induction. We proceed by well-founded induction over $\langle \mathcal{P}, \mathbb{N} \rangle$, equipped with the lexicographic product of $\overset{\tau}{\Rightarrow} \cdot S^+$ and the standard ordering of natural numbers, which are two well-founded relations (the termination of $\overset{\tau}{\Rightarrow} \cdot S^+$ is equivalent to that of $S^+ \cdot \overset{\tau}{\Rightarrow}$). We use the predicate $\varphi(u, n)$:

"for any p, p'_0, q, $u \to^\star p \to^n \cdot \hookrightarrow p'_0$ and $p\ R\ q$ entail $p'_0\ R^\star \cdot \hookleftarrow \cdot \leftarrow^\star q$."

- if $n = 0$, then $\varphi(u, n)$ holds by using the commutation hypothesis (H');
- otherwise, take p_0 such that $p \to p_0 \to^{n-1} \cdot \hookrightarrow p'_0$, and apply the first commutation hypothesis (H) to $p_0 \hookleftarrow \cdot \leftarrow p\ R\ q$: there exist $k > 0$ and $p_1, \ldots, p_k$ such that $q \to^\star \cdot \hookrightarrow p_k$, $p_{k-1}\ R\ p_k$ and $\forall i \in [1; k-1],\ p_{i-1}\ S\ p_i$. We now define by an internal induction a sequence $(p'_i)_{0<i\leq k}$ such that we have $\forall i \in [1; k],\ p_{i-1}\ R^\star\ p'_i \hookleftarrow \cdot \leftarrow^\star p_i$.
  - if $i = 1$, we apply the external induction hypothesis: $\varphi(u, n-1)$, to $p'_0 \hookleftarrow \cdot \leftarrow^{n-1} p_0\ R\ p_1$ (recall that $S \subseteq R$): there exists p'_1 such that $p'_0\ R^\star\ p'_1$ and $p_1 \to^\star \cdot \hookrightarrow p'_1$.
  - otherwise, $i > 1$, we suppose that the sequence is constructed until $i-1$, and we remark that $u \to^+ \cdot\ S^+\ p_{i-1}$, so that we can obtain p'_i by applying the external induction hypothesis, $\varphi(p_{i-1}, m_{i-1})$, to $p'_{i-1} \hookleftarrow \cdot \leftarrow^{m_{i-1}} p_{i-1}\ R\ p_i$ (m_{i-1} is the number of steps between p_{i-1} and p'_{i-1}).

  We can conclude: we have $p'_0\ R^\star\ p'_k \hookleftarrow \cdot \leftarrow^\star q$.

  This case of the proof is summed up below in a diagrammatic way:



■

Theorem 4.2. *Let R, S be two relations; suppose that $S^+ \cdot \overset{\tau}{\Rightarrow}$ terminates.*

$$\textit{If } S \subseteq R \textit{ and } \begin{cases} \overset{\tau}{\leftarrow} \cdot\, R \subseteq S^\star \cdot R \cdot \overset{\widehat{\tau}}{\Leftarrow} \\ \forall a \in \mathcal{L}^{\mathrm{v}},\ \overset{a}{\leftarrow} \cdot\, R \subseteq R^\star \cdot \overset{\widehat{a}}{\Leftarrow} \end{cases} \textit{ then } \quad R^\star \textit{ is a } \mathbf{w}\textit{-simulation.}$$

Proof. We first apply Lemma 4.1 with $\to = \overset{\tau}{\to}$ and $\hookrightarrow = I$, so that we obtain $\overset{\widehat{\tau}}{\Leftarrow} \cdot R^\star \subseteq R^\star \cdot \overset{\widehat{\tau}}{\Leftarrow}$.

This leads to $\overset{\widehat{\tau}}{\Leftarrow} \cdot \overset{a}{\leftarrow} \cdot R \subseteq R^\star \cdot \overset{\widehat{a}}{\Leftarrow}$, so that we can apply Lemma 4.1 again, with $\to = \overset{\tau}{\to}$ and $\hookrightarrow = \overset{a}{\to} \cdot \overset{\widehat{\tau}}{\Rightarrow}$, to obtain $\overset{\widehat{a}}{\Leftarrow} \cdot R^\star \subseteq R^\star \cdot \overset{\widehat{a}}{\Leftarrow}$. ■

The proof is given in appendix; intuitively, this theorem allows reasoning up to transitivity, provided that the pairs used in transitivity position in silent challenges (those collected in relation S) satisfy a termination property. Restricted to the case $R = S \cup I$, this corresponds to [16, Thm. 3.13]. This generalisation, which may seem useless, makes the result much more tractable in practise: the termination requirement refers only to the part of R that is actually used in silent challenges, to rewrite the left-hand-side process. Therefore, we can enlarge R according to our need, without having to bother with the termination of $S^+ \cdot \stackrel{\tau}{\Rightarrow}$. Notably, and unlike in [16], $S^\star$ is not required to be a $\mathbf{w}$-simulation by itself. Also remark that the termination requirement does not entail the termination of S or $\stackrel{\tau}{\rightarrow}$, which makes it realistic in practise. An application, where this kind of requirement comes from the termination of $\stackrel{\tau}{\rightarrow}$ and the fact that S does not interfere with the termination argument is described in [13].

In order to integrate this technique into our setting, we have to define a map that enforces the termination hypothesis. We achieve this by using an external relation that will satisfy the termination hypothesis: let $\succ$ be a transitive relation and define $t_\succ : R \mapsto (R \cap \succ)^\star \cdot R$.

Corollary 4.3. *If $\succ \cdot \stackrel{\tau}{\Rightarrow}$ terminates, then $t_\succ$ is $\mathbf{w}$- and $\mathbf{w}_t$-correct via $\mathrm{id}^\star_{\mathcal{R}}$.*

Proof. Given a $(\mathbf{w}_\mathrm{t} \circ t_\succ)$-simulation R, we apply Thm. 4.2 to R and $S = R \cap \succ$: $R^\star$ is a $\mathbf{w}$-simulation. Furthermore, $\mathbf{w}_\mathrm{t}$-correctness via $\mathrm{id}^\star_{\mathcal{R}}$ entails $\mathbf{w}$-correctness $\mathrm{id}^\star_{\mathcal{R}}$. ■

It, then suffices to establish the following (elementary) properties, so that we can combine this correct map with standard compatible maps, using Thm. 1.8.

Lemma 4.4. *Let $\mathcal{C}$ be a closure such that $\mathcal{C}(\succ) \subseteq \succ$, let S be a reflexive relation. The maps $\mathcal{C}$, $R \mapsto S$ and $R \mapsto R \cdot S$ are $t_\succ$-compatible.*

Proof. – $\mathcal{C}(R \cap \succ) \subseteq \mathcal{C}(R) \cap \mathcal{C}(\succ) \subseteq \mathcal{C}(R) \cap \succ$, and $\mathcal{C}$ is a closure, hence:

$$\begin{aligned} \mathcal{C} \circ t_\succ(R) = \mathcal{C}\left((R \cap \succ)^\star \cdot R\right) &\subseteq (\mathcal{C}(R \cap \succ))^\star \cdot \mathcal{C}(R) \\ &\subseteq (\mathcal{C}(R) \cap \succ)^\star \cdot \mathcal{C}(R) = t_\succ \circ \mathcal{C}(R) \ ; \end{aligned}$$

– $\widehat{S} \circ t_\succ(R) = S \subseteq (S \cap \succ)^\star \cdot S = t_\succ(S) = t_\succ \circ \widehat{S}(R)$.
– S being reflexive, we have $R \subseteq R \cdot S$, so that $t_\succ(R) \cdot S = (R \cap \succ)^\star \cdot R \cdot S \subseteq ((R \cdot S) \cap \succ)^\star \cdot R \cdot S = t_\succ(R \cdot S)$. ■

Theorem 4.5. *Let $\mathcal{C}$ be a $\mathbf{w}$-compatible closure such that $\mathcal{C}(\succ) \subseteq \succ$. If $\succ \cdot \stackrel{\tau}{\Rightarrow}$ terminates, $R \mapsto ((\mathcal{C}(R) \cup \approx) \cap \succ)^\star \cdot \mathcal{C}(R) \cdot \approx$ is $\mathbf{w}$-correct via a symmetric map.*

Proof. Let $f : R \mapsto (\mathcal{C}(R) \cup \approx) \cdot \approx$. Using Props. 1.6, 2.3, Cor. 4.3, Lemma 4.4 and Thm. 1.8, $t_\succ \circ f$ is $\mathbf{w}$-correct via $(f^\omega)^\star = (\mathcal{C}^\omega \cup \widehat{\approx})^\star$, which is symmetric. Then, we have $((\mathcal{C}(R) \cup \approx) \cap \succ)^\star \cdot \mathcal{C}(R) \cdot \approx \subseteq t_\succ \circ f(R)$. ■

This theorem also holds for $\mathbf{w}_\mathrm{t}$; it is however unclear whether there are interesting $\mathbf{w}_\mathrm{t}$-compatible closures, as explained in Sect. 3. We conclude by considering *elaboration* ($\succsim$) [2], which is another coinductively defined preorder contained

in $\approx$. We have shown in [14] that this preorder can be used as an up-to technique for $\approx$, when $\xrightarrow{\tau}$ terminates. Using our theory, we can combine this result with up to context: if $\xrightarrow{\tau}$ terminates, so does $\gtrsim \cdot \stackrel{\tau}{\Rightarrow}$ [14, Lemma 2.5]; we can moreover show that elaboration is a congruence w.r.t CCS contexts, so that $\gtrsim$ naturally satisfies the requirements of Thm. 4.5. Notice that we could encompass up-to techniques for elaboration in our theory, using similar arguments as for expansion.

Corollary 4.6. *In finite (replication free) CCS, map* $R \mapsto \gtrsim \cdot \mathcal{C}_{ccs}(R) \cdot \approx$ *is* **w***-correct via a symmetric map.*

Proof. We have $((\mathcal{C}_{ccs}(R) \cup \approx) \cap \gtrsim)^\star \cdot \mathcal{C}_{ccs}(R) \cdot \approx \; = \; \gtrsim \cdot \mathcal{C}_{ccs}(R) \cdot \approx$. ■

5 Related and Future Work

Termination in the point-free setting. We would like to investigate whether the presentation of the techniques exploiting termination arguments and well-founded induction (Sect. 4) can be lifted to the point-free setting of Sect. 2. Results from [4], in the setting of *relation algebras*, are really encouraging: terminating relations can be characterised at a point-free level, and this property can be related to corresponding well-founded induction principles. Notably, Newman's Lemma, whose proof uses the same ingredients as our proof of Lemma 4.1 (e.g., diagram chasing and well-founded induction), can be proved at the corresponding abstraction level. Relation algebras are slightly more restrictive than our setting however: they require a completely distributive complete lattice (e.g., that arbitrary lubs distribute over arbitrary glbs) and a "modular identity law".

Termination and contexts. In order to use Théorème. 4.5 with a closure ($\mathcal{C}$), we have to check that relation $\succ$, which ensures the termination requirement ($\succ \cdot \stackrel{\tau}{\Rightarrow}$), is closed under $\mathcal{C}$ ($\mathcal{C}(\succ) \subseteq \succ$). This hypothesis is automatically satisfied by elaboration, which is a pre-congruence; however, we would like to investigate more generally how to obtain such pre-congruences satisfying the termination requirement. This is a common question in rewriting theory; we plan to study whether tools from this domain (rewrite orders, dependency pairs, interpretations) can be adapted to our case, where the termination property is about the composition of the relation with silent transitions, rather than about the relation itself.

Congruence properties. In the case of sum-free CCS, which we studied in Sect. 3, bisimilarities are congruences w.r.t all contexts. Such situations are not so common in concurrency theory, where we often have to close bisimilarity under some contexts, in order to obtain a congruence [21,7]. Our setting seems well-suited to analyse such situations at a rather abstract level: given a closure $\mathcal{C}$, representing the congruence property to be satisfied, we can define its adjoint as the map $\mathcal{C}^\circ : x \mapsto \bigvee \{y \mid \mathcal{C}(y) \sqsubseteq x\}$. We have $\mathcal{C}^\circ \circ \mathcal{C} = \mathcal{C}$, $\mathcal{C} \circ \mathcal{C}^\circ = \mathcal{C}^\circ$, so that

$\mathcal{C}(x) \sqsubseteq y$ iff $x \sqsubseteq \mathcal{C}^\circ(y)$; therefore, $\mathcal{C}^\circ$ maps any element x to the largest congruence dominated by x. For example, $\mathcal{C}^\circ(\boldsymbol{\nu}\overleftrightarrow{\mathbf{w}})$ is the largest congruence contained in weak bisimilarity. Another standard approach consists in closing the relation under contexts, after each step of the bisimulation games; in doing so, we obtain *barbed congruence* [9,7], which is both a congruence, and a bisimulation. We can capture this approach by considering $\boldsymbol{\nu}\left(\overleftrightarrow{\mathbf{w}} \wedge \mathcal{C}^\circ\right)$. We would like to study whether up-to techniques can be developed in order to reduce the number of contexts to be considered in such cases, and to have a better understanding of the interactions between "game maps" like $\mathbf{w}$ and "congruent maps" like $\mathcal{C}^\circ$.

Acknowledgements. The author is very grateful to Daniel Hirschkoff and Davide Sangiorgi for helpful discussions and suggestions. Moreover, he would like to acknowledge Tom Hirschowitz for an initial idea which lead to Thm. 1.8.

References

1. S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29(9):737–760, 1992.
2. S. Arun-Kumar and V. Natarajan. Conformance: A precongruence close to bisimilarity. In *Proc. Struct. in Concurrency Theory*. Springer Verlag, 1995.
3. B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.
4. H. Doornbos, R. Backhouse, and J. van der Woude. A calculational approach to mathematical induction. *Theoretical Computer Science*, 179(1–2):103–135, 1997.
5. C. Fournet. *The Join-Calculus: a Calculus for Distributed Mobile Programming*. PhD thesis, Ecole Polytechnique, 1998.
6. C. Fournet, J.-J. Lévy, and A. Schmitt. An asynchronous, distributed implementation of mobile ambients. In *Proc. IFIP TCS'00*, volume 1872 of *LNCS*, pages 348–364. Springer Verlag, 2000.
7. Cédric Fournet and Georges Gonthier. A hierarchy of equivalences for asynchronous calculi. *Journal of Logic and Algebraic Programming*, 63(1):131–173, 2005.
8. M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. *Math. Struct. in Computer Science*, 14(5):651–684, 2004.
9. Kohei Honda and Nobuka Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151(2):437–486, 1995.
10. D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124:103–112, 1996.
11. S. B. Lassen. Relational reasoning about contexts. In A. D. Gordon and A. M. Pitts, editors, *Higher Order Operational Techniques in Semantics*. Cambridge University Press, 1998.
12. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
13. D. Pous. On bisimulation proofs for the analysis of distributed abstract machines. In *Proc. TGC '06*, volume 4661 of *LNCS*. Springer Verlag, 2006. (to appear).
14. D. Pous. Weak bisimulation up to elaboration. In *Proc. CONCUR '06*, volume 4137 of *LNCS*, pages 390–405. Springer Verlag, 2006.
15. D. Pous. Complete lattices and up-to techniques. In *Proc. APLAS '07*, volume 4807 of *LNCS*, pages 351–366. Springer Verlag, 2007.

16. D. Pous. New up-to techniques for weak bisimulation. *Theoretical Computer Science*, 2007. (an extended abstract appeared in Proc. ICALP '05, LNCS 3580).
17. D. Sangiorgi. On the bisimulation proof method. *Journal of Math. Struct. in Computer Science*, 8:447–479, 1998.
18. D. Sangiorgi. Personal communication, 2006.
19. D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *Proc. LICS '07*, pages 293–302. IEEE Computer Society, 2007.
20. D. Sangiorgi and R. Milner. The problem of "weak bisimulation up to". In *Proc. 3rd CONCUR*, volume 630 of *LNCS*, pages 32–46. Springer Verlag, 1992.
21. D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
22. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, June 1955.