

Complete Lattices and Up-To Techniques^{*}

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 [11] to functional programming [9,10,17].

This approach relies on Knaster-Tarski’s fixpoint theorem [20]: “*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 [11,15], 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 [10,19,5,12,17], and turn out to be essential in some cases.

In this paper, we generalise the theory of [15] to the abstract setting of complete lattices [3]. This allows us to ignore the technicalities of LTSs and binary

^{*} This work has been supported by the french project “ModyFiable”, funded by ANR ARASSIA.

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 [15]. 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 [12] 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* [1,18]); we have to resort to recent, and more sophisticated techniques [14] relying on termination hypotheses.

The powerful techniques of [14] 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 [15], where only compatible maps are considered, and [14], 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 [14], 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 $(\overset{\alpha}{\rightarrow})_{\alpha \in \mathcal{L}}$ of objects, indexed by some *labels*, and strong similarity is the largest object x such that the semi-commutation diagram (S) below is satisfied:

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

There is an implicit universal quantification on all labels α , so that this diagram should be read $(S) : \forall \alpha \in \mathcal{L}, \overset{\alpha}{\leftarrow} \cdot x \sqsubseteq x \cdot \overset{\alpha}{\leftarrow}$ (where (\cdot) is the law of the monoid, \sqsubseteq is the partial order of the complete lattice, and $\overset{\alpha}{\leftarrow}$ denotes the converse of relation $\overset{\alpha}{\rightarrow}$). 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* [19,7], where processes are related at a given type and/or in a given typing environment; and *environment bisimulations* [17], 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 [19], and quite hard for functional languages [9,10,17]. 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 $\bigvee Y$. A function $f : X \rightarrow X$ 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: $\text{id}_X : x \mapsto x$; $\hat{y} : x \mapsto y$; $f \circ g : x \mapsto f(g(x))$ and $f^\omega \triangleq \bigvee \{f^n \mid n \in \mathbb{N}\}$, where $f^0 \triangleq \text{id}_X$ and $f^{n+1} \triangleq f \circ f^n$. We say that a map f is *extensive* if $\text{id}_X \sqsubseteq f$.

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 (νs) is the lub of this set:

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

Theorem 1.2 (Knaster-Tarski [20]). νs is the greatest fixpoint of s : $\nu s = s(\nu s)$.

1.2 Up-To Techniques for Proofs by Coinduction

The previous definition gives the powerful *coinduction* proof method: in order to prove that $y \sqsubseteq \nu s$, 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
- $\nu s' \sqsubseteq \nu s$ 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 $\nu(s \circ f) \sqsubseteq \nu s$.

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.

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. 1.10 and Thm. 1.12 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 , $f(\nu s) \sqsubseteq \nu s$. Hence s -compatible maps necessarily correspond to closure properties satisfied by νs . This is not a sufficient condition: there are maps satisfying $f(\nu s) \sqsubseteq \nu s$ that are not s -correct.

Proposition 1.6. The family of s -compatible maps is stable under composition and lubs. It contains the identity, and constant maps \hat{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 $t = \bigvee \{t \mid \nu t \sqsubseteq \nu s\}$ does not necessarily satisfy $\nu t \sqsubseteq \nu s$). 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 [15] (this claim is justified by Sect. 1.4). Thm 1.7, which is our first improvement against [15], allows one to compose correct and compatible maps:

Theorem 1.7. 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 $(g' \circ f^\omega)$.

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 greatest lower bounds (glb): for any $Y \subseteq X$, $\bigwedge Y \triangleq \bigvee \{x \in X \mid \forall y \in Y, x \sqsubseteq y\}$. We extend this definition pointwise to maps. 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 1.8. *We have $X_{\bigwedge S} = \bigcap \{X_s \mid s \in S\}$ and $\nu \bigwedge S \sqsubseteq \bigwedge \{\nu s \mid s \in S\}$.*

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

Proposition 1.9. *Any map that is s -compatible for all s in S is $\bigwedge S$ -compatible.*

Prop. 1.9 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:

Proposition 1.10. *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 $\nu \bigwedge S_f \sqsubseteq \nu \bigwedge S$.

Although Prop. 1.10 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 ($\forall x, \bar{\bar{x}} = x$). For any map f , we define $\bar{f} \triangleq \bar{\cdot} \circ f \circ \bar{\cdot} : x \mapsto \bar{f}(\bar{x})$, and $\overleftarrow{f} \triangleq f \wedge \bar{f}$. We call \bar{x} the *converse* of x and we say that an element x (resp. a map f) is *symmetric* if $x = \bar{x}$ (resp. $f = \bar{f}$). These definitions yield nice algebraic properties (the key point being that we have $x \sqsubseteq y \Leftrightarrow \bar{x} \sqsubseteq \bar{y}$) and we can relate up-to techniques for s and \bar{s} :

Proposition 1.11. *We have $\overline{X_s} = X_{\bar{s}}$, $\overline{\nu s} = \nu \bar{s}$ and for any maps f, f' ,*

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

We can finally combine these properties with Prop. 1.10 and reduce the problem of finding up-to techniques for \overleftarrow{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.

Theorem 1.12. *For any s -correct map f via a symmetric map, $\nu s \circ \overleftarrow{f} \sqsubseteq \nu \overleftarrow{s}$.*

Corollary 1.13. *Let f be an s -correct map via a symmetric map.*

If x is symmetric, and $x \sqsubseteq s(f(x))$, then $x \sqsubseteq \nu \overleftarrow{s}$.

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:

$$\forall x, y, x', y' \in X, x \sqsubseteq x' \wedge y \sqsubseteq y' \Rightarrow x \cdot y \sqsubseteq x' \cdot y' .$$

The *iteration* (resp. *strict iteration*) of an element x is defined by $x^* \triangleq \bigvee_{n \in \mathbb{N}} x^n$ (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: $f \hat{\cdot} g : x \mapsto f(x) \cdot g(x)$, and $f^* : x \mapsto f(x)^*$.

Definition 1.14. 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

$$\forall x, y \in X, s(x) \cdot s(y) \sqsubseteq s(x \cdot y) .$$

Proposition 1.15. *If s preserves the monoid, then:*

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

1.4 Progressions

While maps and pre-fixpoints are the adequate tool in order to build the previous theory of up-to techniques, it is more convenient in practise to use the following notion of *progression*, which can systematically be turned into a map. This notion facilitates the definition of maps corresponding to the various behavioural preorders we will consider in Sect. 2; moreover, it leads to the important results given in Sect. 1.5 about up-to techniques for compatible maps.

Definition 1.16. A *progression* is a binary relation $\succ \subseteq X \times X$, such that:

$$\begin{aligned} \forall x, x', y', y \in X, x \sqsubseteq x' \wedge x' \succ y' \wedge y' \sqsubseteq y \Rightarrow x \succ y , \\ \forall Y \subseteq X, \forall z \in X, (\forall y \in Y, y \succ z) \Rightarrow \bigvee Y \succ z . \end{aligned}$$

We associate to such relation the map $s_{\succ} : x \mapsto \bigvee \{y \in X \mid y \succ x\}$.

Relations \succ in [15], and \rightsquigarrow in [19] are particular instances of progressions. The main advantages of progressions are the following characterisations of the previous notions:

Proposition 1.17. *For any progression \succ , we have:*

- (i) $\forall x, y \in X, x \sqsubseteq s_{\succ}(y) \Leftrightarrow x \succ y$, and in particular, $x \in X_{s_{\succ}}$ iff $x \succ x$;
- (ii) a map f is s_{\succ} -compatible iff $\forall x, y \in X, x \succ y \Rightarrow f(x) \succ f(y)$.
- (iii) s_{\succ} preserves the monoid iff $e \succ e$ and

$$\forall x, y, x', y' \in X, x \succ x' \wedge y \succ y' \Rightarrow x \cdot y \succ x' \cdot y' .$$

Another practical consequence of (i) is that $\forall x, x \sqsubseteq s_{\succ}(f(x))$ iff $x \succ f(x)$.

1.5 Up-To Techniques for Compatible Maps

Denoting by $X^{\langle X \rangle}$ the set of (order-preserving) maps over X , $\langle X^{\langle X \rangle}, \sqsubseteq, \bigvee, \circ, \text{id}_X \rangle$ 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:

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

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 ($s = s_{\mapsto}$), we have $f \overset{s}{\rightsquigarrow} f'$ iff $\forall x, y \in X, x \mapsto y \Rightarrow f(x) \mapsto f'(y)$.

Lemma 1.18. *For any map s , $\overset{s}{\rightsquigarrow}$ preserves the monoid $\langle X^{\langle X \rangle}, \circ, \text{id}_X \rangle$.*

Theorem 1.19. *Let f, g be two maps.*

- (i) *If the product (\cdot) preserves s and $f \overset{s}{\rightsquigarrow} f^*$, then f^* 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 $(\widehat{\circ})$ is the pointwise extension of (\circ) to functors.

- (i) By Lemma 1.18 and Prop. 1.15, $\varphi = \widehat{\text{id}_X^*} \widehat{\circ} \text{id}_{X^{\langle X \rangle}} : f \mapsto f^*$ is respectful, being the product of two respectful functors:
 - the constant functor to id_X^* , this map being s -compatible by Prop. 1.15;
 - and the identity functor $\text{id}_{X^{\langle X \rangle}}$, 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 $\varphi^\omega(f) = f^*$.
- (ii) By Lemma 1.18 and Prop. 1.15, the functor $\omega \triangleq \text{id}_{X^{\langle X \rangle}}^* : f \mapsto f^\omega$ 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, $\varphi = \widehat{g} \widehat{\circ} \omega : f \mapsto g \circ f^\omega$ is respectful, and $\varphi^\omega(f)$ is s -compatible. We finally check that $\varphi^\omega(f) = g \circ f^\omega$. ■

The first point generalises [19, 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 $\forall x, y, \overline{x \cdot y} = \overline{y} \cdot \overline{x}$.

$$\begin{array}{ll}
\mathbf{s} : & x \mapsto_{\mathbf{s}} y \quad \text{if } x \sqsubseteq y \text{ and } \forall \alpha \in \mathcal{L}, \overleftarrow{\alpha} \cdot x \sqsubseteq y \cdot \overrightarrow{\alpha} ; \\
\mathbf{e} : & x \mapsto_{\mathbf{e}} y \quad \text{if } x \sqsubseteq y \text{ and } \forall \alpha \in \mathcal{L}, \overleftarrow{\alpha} \cdot x \sqsubseteq y \cdot \widehat{\alpha} ; \\
\mathbf{w} : & x \mapsto_{\mathbf{w}} y \quad \text{if } x \sqsubseteq y \text{ and } \forall \alpha \in \mathcal{L}, \overleftarrow{\alpha} \cdot x \sqsubseteq y \cdot \widehat{\alpha} ; \\
\mathbf{w}_t : & x \mapsto_{\mathbf{w}_t} y \quad \text{if } x \sqsubseteq y, \overleftarrow{\tau} \cdot x \sqsubseteq y \cdot \widehat{\tau}, \text{ and } \forall a \in \mathcal{L}^\vee, \overleftarrow{a} \cdot x \sqsubseteq y \cdot \widehat{a} .
\end{array}$$

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

Although we denote by $x, y \dots$ 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 $(\overrightarrow{\alpha})_{\alpha \in \mathcal{L}}$ of relations indexed by \mathcal{L} . Intuitively, $\overrightarrow{\alpha}$ 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 $\mathcal{L}^\vee \triangleq \mathcal{L} \setminus \{\tau\}$, called *visible labels*. For $\alpha \in \mathcal{L}$ we define the following *weak transition relations*:

$$\widehat{\alpha} \triangleq \begin{cases} \overrightarrow{\tau} \vee e & \text{if } \alpha = \tau , \\ \overrightarrow{\alpha} & \text{otherwise ;} \end{cases} \quad \overrightarrow{\alpha} \triangleq \overrightarrow{\tau}^* \cdot \overrightarrow{\alpha} \cdot \overrightarrow{\tau}^* ; \quad \widehat{\alpha} \triangleq \overrightarrow{\tau}^* \cdot \widehat{\alpha} \cdot \overrightarrow{\tau}^* .$$

Notice the following properties: $\widehat{\tau} = \overrightarrow{\tau}^*$, $\overrightarrow{\tau} = \overrightarrow{\tau}^+$, $\widehat{\alpha} = \overrightarrow{\alpha}$. 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: \mathbf{s} yields strong simulation games, where actions are exactly matched (diagram (S) in the introduction); \mathbf{e} yields games corresponding to the left-to-right part of an *expansion* [1,18] game, where it is allowed not to move on silent challenges; and \mathbf{w} yields weak simulation games, where one can answer “modulo silent transitions”. The map (\mathbf{w}_t) is a variant of \mathbf{w} , which allows one to answer up to transitivity on visible challenges. We have $\mathbf{s} \sqsubseteq \mathbf{e} \sqsubseteq \mathbf{w} \sqsubseteq \mathbf{w}_t$, so that $X_{\mathbf{s}} \subseteq X_{\mathbf{e}} \subseteq X_{\mathbf{w}} \subseteq X_{\mathbf{w}_t}$, and $\nu \mathbf{s} \subseteq \nu \mathbf{e} \subseteq \nu \mathbf{w} \subseteq \nu \mathbf{w}_t$.

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. 1.15: 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 [18]. 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}_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 2.1.** (i) *The reflexive transitive map id_X^* 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^* is a \mathbf{w} -simulation; $\nu \mathbf{w}_t = \nu \mathbf{w}$.*

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 $\overline{x} \mapsto_{\mathbf{w}} \overline{x}$. 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:

$$\sim \triangleq \nu \overleftarrow{\mathbf{s}} \qquad \succsim \triangleq \nu(\mathbf{e} \wedge \overline{\mathbf{w}}) \qquad \approx \triangleq \nu \overleftarrow{\mathbf{w}}$$

Strong bisimilarity (\sim) and *weak-bisimilarity* (\approx) are symmetric, reflexive and transitive; *expansion* [1,18] (\succsim) is reflexive and transitive; we have $\sim \sqsubseteq \succsim \sqsubseteq \approx$.

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.

Definition 2.2. A *closure* is a continuous, extensive and symmetric map \mathcal{C} , such that $\forall x, y \in X$, $\mathcal{C}(x \cdot y) \sqsubseteq \mathcal{C}(x) \cdot \mathcal{C}(y)$.

Theorem 2.3. *Let \mathcal{C} be a closure.*

- (i) *If \mathcal{C} is \mathbf{s} -compatible, $x \mapsto (\mathcal{C}(x) \vee \sim)^*$ is $\overleftarrow{\mathbf{s}}$ -compatible*
- (ii) *If \mathcal{C} is \mathbf{w} -compatible, $x \mapsto \succsim \cdot \mathcal{C}(x) \cdot \precsim$ is $\overleftarrow{\mathbf{w}}$ -compatible.*
- (iii) *If \mathcal{C} is \mathbf{w} -compatible, $x \mapsto \succsim \cdot \mathcal{C}(x) \cdot \approx$ is \mathbf{w} -correct via a symmetric map.*
- (iv) $\nu \overleftarrow{\mathbf{w}}_t = \approx$.

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 (ii) and slightly improved in (iii); notice that we need for that to use the notion of correct map: this map is not $\overleftarrow{\mathbf{w}}$ -compatible. Technique (iii) appears on the second diagram below. Finally, (iv) allows us to work up to transitivity on visible actions; which is depicted on the last two diagrams below ((iii) holds for \mathbf{w}_t , provided \mathcal{C} is \mathbf{w}_t -compatible, this hypothesis is however problematic, as explained in Sect. 3). We omitted proof techniques for expansion, which can naturally be recovered from up-to techniques for \mathbf{w} and \mathbf{e} using Prop. 1.10.

$$\begin{array}{cccc} \begin{array}{c} \cdot \\ \alpha \downarrow \end{array} & x & \begin{array}{c} \downarrow \alpha \\ \cdot \end{array} & \\ \cdot & \sqsubseteq & \cdot & \\ \alpha \downarrow & & \downarrow \alpha & \\ \cdot & & \cdot & \\ (\mathcal{C}(x) \vee \sim)^* & & & \end{array} \quad \begin{array}{c} \cdot \\ \alpha \downarrow \end{array} \quad \begin{array}{c} x \\ \sqsubseteq \\ \cdot \\ \Downarrow \hat{\alpha} \end{array} \quad \begin{array}{c} \cdot \\ \tau \downarrow \end{array} \quad \begin{array}{c} x \\ \sqsubseteq \\ \cdot \\ \Downarrow \hat{\tau} \end{array} \quad \begin{array}{c} \cdot \\ a \downarrow \end{array} \quad \begin{array}{c} x \\ \sqsubseteq \\ \cdot \\ \Downarrow a \end{array} \\ \cdot & & \cdot & & \cdot & & \cdot & & \cdot \\ \succsim \cdot \mathcal{C}(x) \cdot \precsim & & \succsim \cdot \mathcal{C}(x) \cdot \approx & & \succsim \cdot \mathcal{C}(x) \cdot \approx & & (\mathcal{C}(x) \vee \approx)^* & & \end{array}$$

$$\begin{array}{c}
\alpha \in \mathcal{L} \quad a \in \mathcal{L}^\vee \\
\bar{\tau} = \tau \quad \bar{a} = a \\
p ::= \mathbf{0} \mid \alpha.p \mid p|p \mid (\nu a)p \mid !p
\end{array}
\quad
\frac{p \xrightarrow{\alpha} p'}{p \mid q \xrightarrow{\alpha} p' \mid q} \quad
\frac{q \xrightarrow{\alpha} q'}{p \mid q \xrightarrow{\alpha} p \mid q'} \quad
\frac{p \xrightarrow{\alpha} p' \quad q \xrightarrow{\bar{\alpha}} q'}{p \mid q \xrightarrow{\tau} p' \mid q'}$$

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

Fig. 2. Calculus of Communicating Systems (CCS)

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. We study the case of (sum-free) CCS [11], whose syntax and semantics are recalled in Fig. 2. The sum operator could easily be added; it is omitted here for lack of space. 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 $\bar{R} \triangleq \{\langle p, q \rangle \mid q R p\}$. We finally equip relations with set-theoretic inclusion (\subseteq) and union (\cup), so that $\langle \mathcal{R}, \subseteq, \cup, \cdot, I, \bar{\cdot} \rangle$ forms a monoidal complete lattice with symmetry.

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

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

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

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

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

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

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 [19]: 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 $\mathcal{C}_{ccs} \subseteq [C_m]^*$. 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 1.19(i) is used in [19], so that the proof in the strong case – reformulated into our setting – amounts to proving $[C_m] \xrightarrow{\mathfrak{s}} [C_m]^*$, i.e., $\forall c \in C_m, [c] \xrightarrow{\mathfrak{s}} [C_m]^*$, which is done by structural induction on context c (recall that $f \xrightarrow{\mathfrak{s}} f'$ iff $R \mapsto_{\mathfrak{s}} S$ entail $f(R) \mapsto_{\mathfrak{s}} f'(S)$). This approach does not scale to the weak case however, where up to transitivity is not correct, so that Thm 1.19(i) cannot no longer be used. Therefore, [19] 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 1.19(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 $c[p_1, \dots, p_n]$. 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 \mathfrak{s} -compatible.*

Proof. By Thm.1.19(ii), it suffices to show $[C_i] \xrightarrow{\mathfrak{s}} \mathcal{C}_{ccs}$, i.e., $\forall c \in C_i, [c] \xrightarrow{\mathfrak{s}} \mathcal{C}_{ccs}$. We study each context of C_i separately, and we show

$$\begin{array}{lll} [\text{id}_{\mathcal{P}}] = \text{id}_{\mathcal{R}} \xrightarrow{\mathfrak{s}} \text{id}_{\mathcal{R}} & [\mathbf{0}] \xrightarrow{\mathfrak{s}} [\mathbf{0}] & [\alpha.] \xrightarrow{\mathfrak{s}} \text{id}_{\mathcal{R}} \\ [(\nu a)] \xrightarrow{\mathfrak{s}} [(\nu a)] & [||] \xrightarrow{\mathfrak{s}} [||] & [!] \xrightarrow{\mathfrak{s}} [!]\omega \circ ([!] \cup \text{id}_{\mathcal{R}}) \end{array}$$

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

$\text{id}_{\mathcal{R}}, [\mathbf{0}]$: straightforward.

$[\alpha.]$: $u = \alpha'.p \xrightarrow{\alpha} u', v = \alpha'.q$ with $p R q$. Necessarily, $\alpha = \alpha'$ and $u' = p$.

We hence have $v = \alpha.q \xrightarrow{\alpha} q$, with $p \text{id}_{\mathcal{R}}(S) q$, (recall that $R \mapsto_{\mathfrak{s}} S$ entails $R \subseteq S$).

$[(\nu a)]$: $u = (\nu a)p \xrightarrow{\alpha} u', v = (\nu a)q$ with $p R q$. Inferences rules impose $u' = (\nu a)p'$ where $p \xrightarrow{\alpha} p'$ and $\alpha \neq a, \bar{a}$. Since $p R q$, we obtain q' such that $q \xrightarrow{\alpha} q'$ and $p' S q'$, and we check that $v \xrightarrow{\alpha} v' = (\nu a)q'$, with $u' [(\nu a)](S) v'$.

$[||]$: $u = p_1|p_2 \xrightarrow{\alpha} u', v = q_1|q_2$ 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 $p_1 \xrightarrow{\alpha} p'_1$. Since $R \mapsto_{\mathfrak{s}} S$, $q_1 \xrightarrow{\alpha} q'_1$ with $p'_1 S q'_1$. We check that $v \xrightarrow{\alpha} v' = q'_1|q_2$ and $u' [||](S) v'$ (again we use $R \mapsto_{\mathfrak{s}} S \Rightarrow R \subseteq S$).
- $u' = p_1|p'_2$ with $p_2 \xrightarrow{\alpha} p'_2$, which is identical to the previous case.
- $u' = p'_1|p'_2$ with $p_1 \xrightarrow{\alpha} p'_1, p_2 \xrightarrow{\bar{\alpha}} p'_2$, and $\alpha = \tau$. We have $q_1 \xrightarrow{\alpha} q'_1, q_2 \xrightarrow{\bar{\alpha}} q'_2$ with $p'_1 S q'_1$ and $p'_2 S q'_2$; so that $v \xrightarrow{\tau} v' = q'_1|q'_2$ and $u' [||](S) v'$.

$[!]$: this case is handled in the proof of Thm 3.3 below, so that we omit it here. ■

$$\begin{array}{c}
R = \{ \langle a, (\nu b)(b.a|\bar{b}) \rangle, \langle b.a, b \rangle, \langle (\nu b)(b|\bar{b}), \mathbf{0} \rangle, \langle (\nu b)\mathbf{0}, \mathbf{0} \rangle \} \\
c : p \mapsto (\nu b)(p|\bar{b})
\end{array}
\quad
\begin{array}{c}
\begin{array}{ccccc}
& & b.a & R & b \\
& \swarrow & & & \searrow \\
a & R & c[b.a] & C_{ccs}(R) & c[b] & R & \mathbf{0}
\end{array}
\end{array}$$

Fig. 3. Closure C_{ccs} is not \mathbf{w}_t -correct

Contrarily to what is announced in [19, Lem. 2.4.52], C_{ccs} is not \mathbf{w} -compatible: consider for example $R = \{ \langle \tau.a, a \rangle \} \cup I$; although $R \mapsto_{\mathbf{e}} R$, $C_{ccs}(R) \mapsto_{\mathbf{e}} C_{ccs}(R)$ does not hold: the challenge $!\tau.a|a \xrightarrow{\tau} !\tau.a [!](R) !a$ cannot be answered in $C_{ccs}(R)$ 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. [19] 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. 1.19(iii): we work “up to iteration and a compatible map”.

Theorem 3.3. $R \mapsto \sim \cdot C_{ccs}(R) \cdot \sim$ is an \mathbf{e} - and \mathbf{w} -compatible closure.

Proof (w-compatibility). Take $g : r \mapsto \sim \cdot R \cdot \sim$; g is \mathbf{w} -compatible, extensive and idempotent; moreover, C_{ccs} being \mathbf{s} -compatible, $C_{ccs}(\sim) \subseteq \sim$, and C_{ccs} is g -compatible. Hence, by Thm.1.19(iii), it suffices to show $\forall c \in C_i, [c] \xrightarrow{\mathbf{w}} g \circ C_{ccs}$.

Like previously, $[\mathbf{0}] \xrightarrow{\mathbf{w}} [\mathbf{0}]$, $[!] \xrightarrow{\mathbf{w}} [!]$, $[\alpha.] \xrightarrow{\mathbf{w}} \text{id}_{\mathcal{R}}$, and $[(\nu a)] \xrightarrow{\mathbf{w}} [(\nu a)]$; we detail the case of the replication, for which we need the map g . Consider R, S such that $R \mapsto_{\mathbf{w}} S$, we have to show $[!](R) \mapsto_{\mathbf{w}} \sim \cdot C_{ccs}(S) \cdot \sim$. 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 $R \xrightarrow{\mathbf{w}} S$, we deduce $q \xrightarrow{\hat{\alpha}} q_0$ with $p_0 S q_0$. There are two cases:
 - $q \xrightarrow{\alpha} q_0$, and we check that $!q \xrightarrow{\alpha} q' = !q|q^k|q_0|q^{k'}$, where $p' C_{ccs}(S) q'$.
 - $q = q_0$ (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 $p' C_{ccs}(S) q' \sim !q$.
- $p' = !p|p^k|p_0|p^{k'}|p_1|p^{k''}$ with $p \xrightarrow{\alpha} p_0$ and $p \xrightarrow{\bar{\alpha}} p_1$ ($\alpha = \tau$). Since $R \xrightarrow{\mathbf{w}} S$, we deduce $q \xrightarrow{\alpha} q_0$ and $q \xrightarrow{\bar{\alpha}} q_1$ with $p_0 S q_0$ and $p_1 S q_1$. We check that $!q \xrightarrow{\tau} q' = !q|q^k|q_0|q^{k'}|q_1|q^{k''}$, where $p' C_{ccs}(S) q'$. ■

A Negative Result. Rather surprisingly, C_{ccs} is not \mathbf{w}_t -correct: a counterexample [16] is depicted on Fig. 3, where R is not contained in \mathbf{w}_t -similarity while R is a $(\mathbf{w}_t \circ C_{ccs})$ -simulation. The point is that $[!] \xrightarrow{\mathbf{w}_t} C_{ccs}$ 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}_t , to silent challenges – where it is not

This shows that maps inducing the same fixpoint (recall that $\nu \mathbf{w} = \nu \mathbf{w}_t$) 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 ($\xrightarrow{\hat{\alpha}}$)”: the relation R from Fig. 3 also satisfies $\forall \alpha, \hat{\alpha} \cdot R \subseteq C_{ccs}(R)^* \cdot \hat{\alpha}$.

4 Going Beyond Expansion: Termination Hypotheses

In recent work [14], 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 [12]), 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 $\forall i \in \mathbb{N}, p_i \succ p_{i+1}$.

Theorem 4.1. *Let R, S be two relations; suppose that $S^+ \cdot \xrightarrow{\tau}$ terminates.*

$$\text{If } S \subseteq R \text{ and } \begin{cases} \xrightarrow{\tau} \cdot R \subseteq S^* \cdot R \cdot \xleftarrow{\tau} \\ \forall a \in \mathcal{L}^v, \xrightarrow{a} \cdot R \subseteq R^* \cdot \xleftarrow{a} \end{cases} \text{ then } R^* \text{ is a } \mathbf{w}\text{-simulation.}$$

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 [14, 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 $S^+ \cdot \xrightarrow{\tau}$. Notably, and unlike in [14], S^* 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 $\xrightarrow{\tau}$, which makes it realistic in practise. An application, where this kind of requirement comes from the termination of $\xrightarrow{\tau}$ and the fact that S does not interfere with the termination argument is described in [12].

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 $t_\succ : R \mapsto (R \cap \succ)^* \cdot R$.

Corollary 4.2. *If $\succ \cdot \xrightarrow{\tau}$ terminates, then t_\succ is \mathbf{w} - and \mathbf{w}_t -correct via $\text{id}_{\mathcal{R}}^*$.*

It, then suffices to establish the following (elementary) properties, so that we can combine this correct map with standard compatible maps, using Thm. 1.7.

Lemma 4.3. *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.*

Theorem 4.4. *Let \mathcal{C} be a \mathbf{w} -compatible closure such that $\mathcal{C}(\succ) \subseteq \succ$. If $\succ \cdot \xrightarrow{\tau}$ terminates, $R \mapsto ((\mathcal{C}(R) \cup \approx) \cap \succ)^* \cdot \mathcal{C}(R) \cdot \approx$ is \mathbf{w} -correct via a symmetric map.*

This theorem also holds for \mathbf{w}_t ; it is however unclear whether there are interesting \mathbf{w}_t -compatible closures, as explained in Sect. 3. We conclude by considering *elaboration* (\approx) [2], which is another coinductively defined preorder contained in \approx . We have shown in [13] 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 $\approx \cdot \xrightarrow{\tau}$ [13, Lemma 2.5]; we can moreover show that elaboration is a congruence w.r.t CCS contexts, so that \approx naturally satisfies the requirements of Thm. 4.4.

Corollary 4.5. *In finite (replication free) CCS, map $R \mapsto \approx \cdot \mathcal{C}_{ccs}(R) \cdot \approx$ is w-correct via a symmetric map.*

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 A.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. 4.4 with a closure \mathcal{C} , we have to check that relation \succ , which ensures the termination requirement ($\succ \cdot \xrightarrow{\tau}$), 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 [19,6]. 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}° maps any element x to the largest congruence dominated by x . For example, $\mathcal{C}^\circ(\nu \overleftarrow{\mathbf{w}})$ 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* [8,6], which is both a congruence, and a bisimulation. We can capture this approach by considering $\nu(\overleftarrow{\mathbf{w}} \wedge \mathcal{C}^\circ)$. 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}° .

Acknowledgements. The author is very grateful to Daniel Hirschhoff and Davide Sangiorgi for helpful discussions and suggestions. Moreover, he would like to acknowledge Tom Hirschowitz for an initial idea which lead to Thm. 1.7.

References

1. Arun-Kumar, S., Hennessy, M.: An efficiency preorder for processes. *Acta Informatica* 29(9), 737–760 (1992)
2. Arun-Kumar, S., Natarajan, V.: Conformance: A precongruence close to bisimilarity. In: *Proc. Struct. in Concurrency Theory*, Springer, Heidelberg (1995)
3. Davey, B., Priestley, H.: *Introduction to Lattices and Order*. Cambridge University Press, Cambridge (1990)
4. Doornbos, H., Backhouse, R., van der Woude, J.: A calculational approach to mathematical induction. *Theoretical Computer Science* 179(1–2), 103–135 (1997)
5. Fournet, C., Lévy, J.-J., Schmitt, A.: An asynchronous, distributed implementation of mobile ambients. In: Watanabe, O., Hagiya, M., Ito, T., van Leeuwen, J., Mosses, P.D. (eds.) *TCS 2000. LNCS*, vol. 1872, pp. 348–364. Springer, Heidelberg (2000)
6. Fournet, C., Gonthier, G.: A hierarchy of equivalences for asynchronous calculi. *Journal of Logic and Algebraic Programming* 63(1), 131–173 (2005)
7. Hennessy, M., Rathke, J.: Typed behavioural equivalences for processes in the presence of subtyping. *Math. Struct. in Computer Science* 14(5), 651–684 (2004)
8. Honda, K., Yoshida, N.: Kohei Honda and Nobuka Yoshida. *Theoretical Computer Science* 151(2), 437–486 (1995)
9. Howe, D.J.: Proving congruence of bisimulation in functional programming languages. *Information and Computation* 124, 103–112 (1996)
10. Lassen, S.B.: Relational reasoning about contexts. In: Gordon, A.D., Pitts, A.M. (eds.) *Higher Order Operational Techniques in Semantics*, Cambridge University Press, Cambridge (1998)
11. Milner, R.: *Communication and Concurrency*. Prentice-Hall, Englewood Cliffs (1989)
12. Pous, D.: On bisimulation proofs for the analysis of distributed abstract machines. In: Pous, D. (ed.) *TGC 2006. LNCS*, vol. 4661, Springer (to appear)
13. Pous, D.: Weak bisimulation up to elaboration. In: Baier, C., Hermanns, H. (eds.) *CONCUR 2006. LNCS*, vol. 4137, pp. 390–405. Springer, Heidelberg (2006)
14. Pous, D.: New up-to techniques for weak bisimulation. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) *ICALP 2005. LNCS*, vol. 3580, Springer, Heidelberg (2005)
15. Sangiorgi, D.: On the bisimulation proof method. *Journal of Math. Struct. in Computer Science* 8, 447–479 (1998)
16. Sangiorgi, D.: Personal communication (2006)
17. Sangiorgi, D., Kobayashi, N., Sumii, E.: Environmental bisimulations for higher-order languages. In: *LICS 2007*, pp. 293–302. IEEE Computer Society Press, Los Alamitos (2007)
18. Sangiorgi, D., Milner, R.: The problem of “weak bisimulation up to”. In: Cleaveland, W.R. (ed.) *CONCUR 1992. LNCS*, vol. 630, pp. 32–46. Springer, Heidelberg (1992)
19. Sangiorgi, D., Walker, D.: *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press, Cambridge (2001)
20. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics* 5(2), 285–309 (1955)

A Proof of Theorem 4.1

We give the proof of Thm 4.1; this requires a technical lemma expressing the commutation property on which the technique relies.

Lemma A.1. *Let R, S, \rightarrow and \hookrightarrow be four relations. If $S \subseteq R$, and $S^+ \cdot \rightarrow^+$ terminates, then*

$$\begin{cases} \hookrightarrow \cdot R \subseteq S^* \cdot R \cdot \hookrightarrow^* & \text{(H)} \\ \hookrightarrow \cdot R \subseteq R^* \cdot \hookrightarrow \cdot \hookrightarrow^* & \text{(H')} \end{cases} \text{ entail } \hookrightarrow \cdot \hookrightarrow^* \cdot R^* \subseteq R^* \cdot \hookrightarrow \cdot \hookrightarrow^* .$$

Proof. We actually prove $\hookrightarrow \cdot \hookrightarrow^* \cdot R \subseteq R^* \cdot \hookrightarrow \cdot \hookrightarrow^*$, 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 $\xrightarrow{\tau} \cdot S^+$ and the standard ordering of natural numbers, which are two well-founded relations (the termination of $\xrightarrow{\tau} \cdot S^+$ is equivalent to that of $S^+ \cdot \xrightarrow{\tau}$). We use the predicate $\varphi(u, n)$:

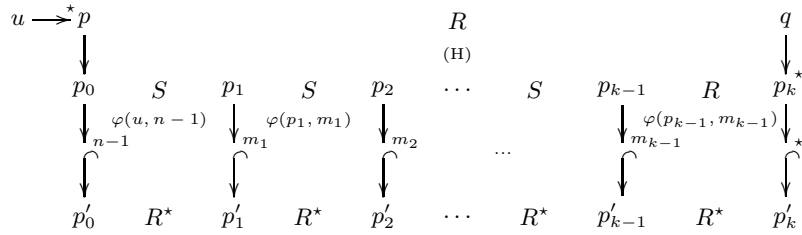
“for any $p, p'_0, q, u \xrightarrow{*} p \xrightarrow{n} \hookrightarrow p'_0$ and $p R q$ entail $p'_0 R^* \cdot \hookrightarrow \cdot \hookrightarrow^* q$.”

- if $n = 0$, then $\varphi(u, n)$ holds by using the commutation hypothesis (H’);
- otherwise, take p_0 such that $p \rightarrow p_0 \xrightarrow{n-1} \hookrightarrow p'_0$, and apply the first commutation hypothesis (H) to $p_0 \hookrightarrow \cdot \hookrightarrow p R q$: there exist $k > 0$ and p_1, \dots, p_k such that $q \xrightarrow{*} \hookrightarrow p_k, p_{k-1} R p_k$ 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 $\forall i \in [1; k], p_{i-1} R^* p'_i \hookrightarrow \cdot \hookrightarrow^* p_i$.

- if $i = 1$, we apply the external induction hypothesis: $\varphi(u, n-1)$, to $p'_0 \hookrightarrow \cdot \hookrightarrow^{n-1} p_0 R p_1$ (recall that $S \subseteq R$): there exists p'_1 such that $p'_0 R^* p'_1$ and $p_1 \xrightarrow{*} \hookrightarrow p'_1$.
- otherwise, $i > 1$, we suppose that the sequence is constructed until $i-1$, and we remark that $u \xrightarrow{+} \cdot S^+ p_{i-1}$, so that we can obtain p'_i by applying the external induction hypothesis, $\varphi(p_{i-1}, m_{i-1})$, to $p'_{i-1} \hookrightarrow \cdot \hookrightarrow^{m_{i-1}} p_{i-1} R p_i$ (m_{i-1} is the number of steps between p_{i-1} and p'_{i-1}).

We can conclude: we have $p'_0 R^* p'_k \hookrightarrow \cdot \hookrightarrow^* q$.

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



■

Proof of Theorem 4.1. We first apply Lemma A.1 with $\rightarrow = \xrightarrow{\tau}$ and $\hookrightarrow = I$, so that we obtain $\xrightarrow{\tau} \cdot R^* \subseteq R^* \cdot \xrightarrow{\tau}$.

This leads to $\xrightarrow{\tau} \cdot \xrightarrow{a} \cdot R \subseteq R^* \cdot \xrightarrow{\hat{a}}$, so that we can apply Lemma A.1 again, with $\rightarrow = \xrightarrow{\tau}$ and $\hookrightarrow = \xrightarrow{a} \cdot \xrightarrow{\hat{a}}$, to obtain $\xrightarrow{\tau} \cdot R^* \subseteq R^* \cdot \xrightarrow{\hat{a}}$. ■