Coinduction All the Way Up

Damien Pous

Coinduction All the Way Up

Damien Pous
Plume team, CNRS, ENS Lyon
Damien.Pous@ens-lyon.fr

Abstract

We revisit coinductive proof principles from a lattice theoretic point of view. By associating to any monotone function a function which we call the companion, we give a new presentation of both Knaster-Tarski's seminal result, and of the more recent theory of enhancements of the coinductive proof method (up-to techniques).

The resulting theory encompasses parametrised coinduction, as recently proposed by Hur et al., and second-order reasoning, i.e., the ability to reason coinductively about the enhancements themselves. It moreover resolves an historical peculiarity about up-to context techniques.

Based on these results, we present an open-ended proof system allowing one to perform proofs on-the-fly and to neatly separate inductive and coinductive phases.

Keywords Coinduction, Enhancements, Complete lattices, GSOS, Parametrised coinduction

1. Introduction

Coinduction is a simple mathematical tool that follows from Knaster-Tarski's fixpoint theorem about complete lattices [14, 32]. It was first used implicitly, for instance in finite automata algorithms [9, 10], until Milner popularised it by proposing *bisimilarity* as a natural way to compare concurrent programs [19]. It was then widely used, for instance to analyse other process-calculi [20], the lambda-calculus [2], cryptographic protocols [1], distributed implementations [8], concurrent ML [13], analytic differential equations [26], or C compilers [17, 29].

The reason for such a success is that like induction, coinduction provides a powerful proof technique: to prove some property by coinduction, it suffices to exhibit an *invariant*. Typically, in program semantics, one can prove the equivalence of two programs by exhibiting a *bisimulation relation* that contains those two programs. The point there is that while program equivalence is a global property (for instance, because equivalent programs should remain equivalent under arbitrary contexts), the conditions for a relation to be a bisimulation are local. By using coinduction one can thus ensure a global property by checking only local properties. Very much like induction allows one to reduce a proof about arbitrary natural numbers to a proof about zero and successor.

From the beginning [19], Milner introduced *enhancements* of the bisimulation proof method. They make it possible to work with relations that are much smaller than actual bisimulations and yet ensure program equivalence: they are always contained in a bisimulation. Those relations are usually called *bisimulations up to*. The benefits of these enhancements can be spectacular; a bisimulation up to can be finite whereas any enclosing bisimulation is infinite. Sometimes it may be hard even *to define* an enclosing bisimulation, let alone carrying out the whole proof. There are many possible enhancements, and they proved useful, if not essential, in proofs about name-passing languages [6, 12, 28], languages with information hiding mechanisms (e.g., existential types, encryption and decryption constructs [1, 30, 31]), and higher-order languages [15, 16].

Sangiorgi developed a first theory of those enhancements [27, 28], which the present author further refined [22–24]. In this line of work, the emphasis was put on compositionality: given the wide variety of enhancements, it is crucial to have tools to analyse each of them separately, and then to combine them when needed in a concrete proof. Since enhancements do not compose in general, Sangiorgi proposed a notion of *respectful* enhancement. These form a subclass of the valid enhancements, and they enjoy nice compositional properties: they are closed under union and composition. One can thus establish a dictionary of respectful enhancements, and then use any combination of those in concrete proofs. In the author's refinement of this framework, respectfulness was modified into *compatibility*, a slightly more natural notion which essentially plays the same role but leads to a smoother theory.

Recently, Hur et al. proposed *parametric coinduction* [11], an extremely neat variation on Knaster-Tarski theorem which allows one to present coinductive proofs incrementally, without having to exhibit the invariant (or bisimulation relation) from the beginning. Such a possibility is especially useful in the context of mechanised formal proofs: the process of discovering the appropriate invariant becomes interactive and amenable to automation.

They also show in this paper how to exploit respectful enhancements with parametric coinduction. When doing so, they define the greatest respectful enhancement and they remark in passing that it is so powerful that there is no point in using a different one.

In the present work, we start from this simple remark and we push it to the limit. Rather than studying respectful or compatible enhancements, we focus on the greatest one from the very beginning. Unexpectedly, doing so leads us to a greatly simplified abstract theory of enhanced coinduction, which encompasses respectfullness, compatibility, parametrised coinduction, and second-order reasoning. More precisely:

- We obtain an alternative proof of Knaster-Tarski's theorem, where compatible enhancements play a central role. Consequently, it becomes straightforward that compatible enhancements can safely be used (Section 3).
- We characterise the greatest compatible enhancement as a greatest fixpoint in the complete lattice of monotone functions. This allows us to reuse our theory of enhanced coinduction at

the second-order level, and to obtain powerful proof techniques for establishing the validity of enhancements. While we already used this approach before [22], focusing on the greatest compatible enhancement brings considerable simplifications and a much cleaner presentation (Section 6).

- Similarly, the formal development of symmetry arguments, which was rather painful to obtain before, gets greatly simplified (Section 7).
- The distinction between respectful and compatible enhancements vanishes in the new theory (Section 9).
- Parametric coinduction, as proposed by Hur et al. becomes a byproduct of the theory. In particular, we derive a simple proof system for parametric enhanced coinduction, where the coinductive phases neatly alternate with the inductive phases corresponding to the enhancements (Section 10).

We illustrate the first-order theory by considering bisimulation proofs in CCS (Section 4) and in Rutten's stream calculus (Section 5). We moreover use the second-order theory to recover up-to-context and congruence results in a compositional and elementary way, for both CCS and the π-calculus (Section 8).

The presented theory and some of the examples have been formalised as a Coq library, which is available from the following webpage: `http://perso.ens-lyon.fr/damien.pous/cawu`

2. Notation and preliminary material

A *complete lattice* is a triple $\langle X, \leqslant, \bigvee \rangle$ where $\langle X, \leqslant \rangle$ is a partial order (reflexive, transitive, and antisymmetric) such that any subset Y of X has a least upper bound $\bigvee Y$: for all $z \in X$,

$$\bigvee Y \leqslant z \qquad \text{iff} \qquad \forall y \in Y,\ y \leqslant z$$

A complete lattice always has a bottom element, written $\bot$, and a binary join operation, written with infix symbol $\vee$:

$$\bot \triangleq \bigvee \emptyset \qquad\qquad x \vee y \triangleq \bigvee \{x, y\}$$

Arbitrary greatest lower bounds can be derived from the least upper bounds. We shall only use binary ones (meets), which we denote with the infix symbol $\wedge$.

Standard examples of complete lattices include: subsets (of a given set) ordered with inclusion; binary relations (on a given set) ordered with inclusion again, and functions into a complete lattice, ordered pointwise. A fourth example, used thoroughly in this paper, is the set of *monotone* functions on a complete lattice.

More precisely, given a complete lattice $\langle X, \leqslant, \bigvee \rangle$, a function $f : X \to X$ is monotone if it preserves the partial order:

$$\forall x, y \in X,\ x \leqslant y \ \Rightarrow f(x) \leqslant f(y)$$

We write $[X \to X]$ for the set of monotone functions on X. When ordered pointwise, this set forms a complete lattice: for all $f, g : [X \to X]$ and $F \subseteq [X \to X]$,

$$f \leqslant g \triangleq \forall x \in X, f(x) \leqslant g(x)$$
$$\bigvee F \triangleq x \mapsto \bigvee_{f \in F} f(x)$$

A *post-fixpoint* of a function $f : [X \to X]$ is an element x such that $x \leqslant f(x)$; a *fixpoint* is an element x such that $x = f(x)$.

In the abstract developments of this paper, we mostly work within a generic complete lattice X, the corresponding lattice of monotone functions $[X \to X]$, and that of monotone functions on $[X \to X]$: $[[X \to X] \to [X \to X]]$. To avoid confusion, we use the following convention: letters x, y, z range over elements of X, letters f, g, b, c, t range over functions in $[X \to X]$, and uppercase letters F, B, S, T are reserved for functions in $[[X \to X] \to [X \to X]]$. We follow the same convention in concrete examples, except that we use bold fonts.

This discipline allows us to overload most symbols in the sequel: for instance, depending on the context, $\bot$ can denote the empty set, the bottom element of an abstract complete lattice X, or the everywhere-bottom function in $[X \to X]$.

To further alleviate notation, we denote the identity function by 1, and both function composition and function application by juxtaposition:

- fx denotes the application of a function f to an element x, usually written $f(x)$;
- fg denotes the composition of two functions f and g, usually written $f \circ g$.

(Similarly for Fg and FB.) We associate juxtapositions to the right when there is no ambiguity. For instance, we write fgx for $f(gx) = (fg)x$, fgb for $f(gb) = (fg)b$, and TTf for $T(Tf) = (TT)f$. In contrast, we keep parentheses in expressions such that $(Bf)g$ and $B(fg)$ which are not equal in general.

3. Knaster-Tarski and Compatibility

Fix a complete lattice $\langle X, \leqslant, \bigvee \rangle$ and pick a function $b : [X \to X]$. Knaster-Tarski's theorem characterises the greatest fixpoint νb of b as the least upper bound of all its post-fixpoints:

$$\nu b = \bigvee_{x \leqslant bx} x \qquad\qquad \frac{x \leqslant y \leqslant by}{x \leqslant \nu b} \tag{1}$$

The corresponding coinduction principle is given on the right-hand side. In words, to prove that x is below in the greatest fixpoint, find a post-fixpoint y above x. The idea of enhancements is to use an additional function f and to look for post-fixpoints of bf rather than b: we switch to the following principle of coinduction up to f

$$\frac{x \leqslant y \leqslant bfy}{x \leqslant \nu b} \tag{2}$$

The function f typically enlarges its argument, and the post-fixpoints of bf can be much smaller than those of b; these are the bisimulations up to we alluded to in the Introduction. The function f corresponds to a valid enhancement when the above rule holds, or, equivalently, when $\nu bf \leqslant \nu b$. Our primary goal is to obtain such functions.

A monotone function $f : [X \to X]$ is *compatible (for b)* if $fb \leqslant bf$. It is straightforward to check that 1 and b are compatible, that the composition of two compatible functions is compatible, and that the least upper bound of a family of compatible functions is compatible.

Definition 3.1. We call *companion of b* the monotone function obtained as the least upper bound of all compatible functions:

$$t \triangleq \bigvee_{fb \leqslant bf} f$$

Lemma 3.2. *The companion is compatible:*

$$tb \leqslant bt \tag{3}$$

Thus this is the greatest compatible function. It moreover satisfies

$$b \leqslant t \tag{4}$$
$$1 \leqslant t \tag{5}$$
$$tt \leqslant t \tag{6}$$

The last two inequalities entail idempotence, i.e., $tt = t$.

Proof. Inequality (3) follows from the fact that supremum of a family of compatible functions is compatible. That it contains b and the identity (4,5) follows from the fact that those functions are compatible. For (6), it suffices to notice that tt is compatible, by (3) and the fact that composition preserves compatibility. □

Our first result is the following alternative definition of the greatest (post-)fixpoint, using the companion:

Theorem 3.3. *The greatest fixpoint of b is the value of the companion on the bottom element.*

$$\nu b = t\bot \tag{7}$$

Proof. We first show that $t\bot$ is the greatest post-fixpoint:

1. $t\bot$ is a post-fixpoint:
$$\begin{aligned} t\bot &\leqslant tb\bot && \text{(monotonicity of } t\text{)} \\ &\leqslant bt\bot && \text{(compatibility of } t\text{)} \end{aligned}$$
2. it is the largest: if $x \leqslant bx$, then the constant-to-x function $\widehat{x}$ is compatible and thus smaller than t, so that $x = \widehat{x}\bot \leqslant t\bot$.

We conclude that $t\bot$ is a fixpoint as in Knaster-Tarski's proof: from monotonicity of b and the first point, $bt\bot$ is also a post-fixpoint, and thus $bt\bot \leqslant t\bot$ by the second point. □

Using idempotence of the companion, we also get

Corollary 3.4. *The companion preserves the greatest fixpoint:*

$$t\nu b = \nu b \tag{8}$$

Typically, when b is the function defining bisimilarity on some process calculus, we recover the fact that if contextual closure is compatible (and thus below t) then bisimilarity is closed under contexts.

Definition 3.5. We call *enhancement of b* the function $b^\dagger \triangleq bt$

This function is an improved version of b, with more post-fixpoints (we have $b \leqslant b^\dagger$) but the same greatest fixpoint:

Theorem 3.6. *The companion is a valid enhancement, we have*

$$\nu b^\dagger = \nu b \tag{9}$$

Proof. From (5) we deduce $b \leqslant b^\dagger$, and thus $\nu b \leqslant \nu b^\dagger$. The interesting result is the other inequality. Since $\nu b^\dagger$ is the greatest post-fixpoint of $b^\dagger$, it suffices to show that any post-fixpoint x of $b^\dagger$ is smaller than νb. We have

$$\begin{aligned} tx \leqslant tb^\dagger x = tbtx \quad & \text{(assumption on } x \text{ and monotonicity of } t\text{)} \\ \leqslant bttx \quad & \text{(t is compatible (3))} \\ \leqslant btx = b^\dagger x \quad & \text{(by (6) and monotonicity of } b\text{)} \end{aligned}$$

Thus tx is a post-fixpoint of b, and $tx \leqslant \nu b$. We conclude with (5): we have $x \leqslant tx \leqslant \nu b$. □

Remark 3.7. In earlier work by Sangiorgi [27] and then by the author [22], where the emphasis was on compatible functions rather that on the greatest one (the companion), the corresponding result is "if f is compatible, and $x \leqslant bfx$ then $x \leqslant \nu b$". Such a result requires a convoluted proof. Indeed, when f is an arbitrary compatible function, one does not have $1 \leqslant f$ and $ff \leqslant f$, and the above proof breaks. Instead, one constructs the sequence $f^0x \triangleq x$, $f^{i+1}x \triangleq ff^ix$ and one shows by recurrence that $f^ix \leqslant bf^{i+1}x$. One deduces that $f^\omega x \triangleq \bigvee_i f^ix$ is a post-fixpoint, so that $x \leqslant f^\omega x \leqslant \nu b$. Focusing on the companion makes it possible to avoid this use of natural numbers.

$$\frac{A \triangleq P \qquad P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'} \qquad \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

Figure 1. Labelled transition system of a fragment of CCS.

Remark 3.8. One might hope to enhance the function b further by using the companion of $b^\dagger$. However we stagnate when doing so: let $t^\star$ be the companion of $b^\dagger$, we have

$$t^\star = t \tag{10}$$

$$b^{\dagger\dagger} = b^\dagger \tag{11}$$

(The proof is given in appendix; note that Equation (10) actually generalises Theorem 3.6: we have $\nu b^\dagger = t^\star\bot = t\bot = \nu b$.)

4. Modularity through the companion

We consider a first example in a fragment of Milner's CCS [19]. Let us recall this process calculus first. We fix a set of *names* $a, b \ldots$, and a set of process constants $A, B, \ldots$. CCS *processes* and *labels* are defined by the following grammar:

$$\begin{aligned} P, Q &::= A \mid 0 \mid \alpha.P \mid P|P \\ \alpha, \beta &::= a \mid \overline{a} \mid \tau \end{aligned}$$

We let $\mathcal{R}, \mathcal{S}$ range over binary relations on processes.

The corresponding labelled transition system (LTS) is given in Figure 1. The first rule accounts for recursion: it assumes that each process constant is associated to a process in some global table. The two symmetrical rule for parallel composition are omitted.

Let $\mathbf{b}$ be the following monotone function on the lattice of binary relations on processes:

$$\begin{aligned} \mathbf{b} : \mathcal{R} \mapsto \{\langle P, Q\rangle \mid \forall \alpha,& \\ & \forall P',\ P \xrightarrow{\alpha} P' \text{ entails } \exists Q',\ Q \xrightarrow{\alpha} Q' \text{ and } P' \mathcal{R} Q' \\ & \forall Q',\ Q \xrightarrow{\alpha} Q' \text{ entails } \exists P',\ P \xrightarrow{\alpha} P' \text{ and } P' \mathcal{R} Q' \} \end{aligned}$$

The so-called *bisimulations* are the post-fixpoints of $\mathbf{b}$, and *bisimilarity* ($\sim$) is its greatest-fixpoint.

An enhanced coinductive proof. Consider the following process definitions, and let us try to prove that $A \sim B$.

$$\begin{aligned} A &\triangleq a.b.D & B &\triangleq a.b.C \\ C &\triangleq \overline{a}.(A|C) & D &\triangleq \overline{a}.(B|D) \end{aligned}$$

Any bisimulation containing the pair $\langle A, B\rangle$ must be infinite. Instead, the companion of $\mathbf{b}$ allows us work with the finite relation $\mathcal{S} \triangleq \{\langle A, B\rangle, \langle C, D\rangle\}$: we have $\mathcal{S} \leqslant \mathbf{b}^\dagger\mathcal{S}$.

Indeed, we have $A \xrightarrow{a} b.D$ and $B \xrightarrow{a} b.C$. While the pair $\langle b.D, b.C\rangle$ does not belong to $\mathcal{S}$, we can use the following functions to cancel the b prefixes and to transpose C and D:

$$\begin{aligned} \mathbf{c}^\cdot &: \mathcal{R} \mapsto \{\langle \alpha.P,\ \alpha.Q\rangle \mid \alpha \text{ a label}, P \mathcal{R} Q\} \\ \mathbf{i} &: \mathcal{R} \mapsto \{\langle Q,\ P\rangle \mid P \mathcal{R} Q\} \end{aligned}$$

We thus have $\langle A, B\rangle \in \mathbf{bic}^\cdot\mathcal{S}$. The function $\mathbf{i}$ is trivially compatible for $\mathbf{b}$, and we shall see in Section 8.1 that $\mathbf{c}^\cdot$ is below the companion of $\mathbf{b}$, written $\mathbf{t}$ in the sequel. Whence $\mathbf{ic}^\cdot \leqslant \mathbf{tt} \leqslant \mathbf{t}$, and thus $\langle A, B\rangle \in \mathbf{b}^\dagger\mathcal{S}$.

Similarly, we have $C \xrightarrow{\overline{a}} A|C$ and $D \xrightarrow{\overline{a}} B|D$, and we use the following function to cancel parallel composition and recover the

two pairs from $\mathcal{S}$:

$$\mathbf{c}^{|} : \mathcal{R} \mapsto \left\{ \langle P|P',\ Q|Q' \rangle \mid P\ \mathcal{R}\ Q, P'\ \mathcal{R}\ Q' \right\}$$

This function is below $\mathbf{t}$ (see Section 8.1 again), so that we get $\langle C, D \rangle \in \mathbf{bc}^{|}\mathcal{S} \leqslant \mathbf{b}^{\dagger}\mathcal{S}$.

Note that thanks to companion, we only had to study transitions along labels a and $\overline{a}$, even though the processes at hand also perform transitions labelled b and τ. (For instance, we have $A \xrightarrow{ab\overline{a}\tau} b.C|B|D$). The fact that the starting processes cannot diverge one from the other using those actions is somehow factored out once and for all, in the proofs that $\mathbf{c}^{\cdot}$ and $\mathbf{c}^{|}$ are valid enhancements.

Modularity. In this example, working with the companion rather than with specific compatible functions is quite convenient: it does not require us to announce a global up-to technique up-front. (Here, something like $\mathbf{ic}^{\cdot} \vee \mathbf{c}^{|}$.) In each sub-case of the proof, we can just extract from the companion whatever is needed for that case. This approach is much more robust, especially in the context of computer-assisted proofs. Suppose for instance that one slightly changes the definition of D into $b.(D|B)$. One can still conclude by reasoning up to commutativity of parallel composition, and this additional technique is already available in the companion: there is no need to update the declared up-to technique, one just needs to adjust the proof locally. (Of course one needs to prove that this new kind of enhancement is available in the companion, but this can be done separately, and once and for all.)

Code reuse. Although this was not needed in the previous example, one can also show that the following function is compatible:

$$\mathbf{j} :\ \mathcal{R} \mapsto \{ \langle P,\ R \rangle \mid \exists Q,\ P\ \mathcal{R}\ Q,\ Q\ \mathcal{R}\ R \}$$

Thus $\mathbf{j} \leqslant \mathbf{t}$, and together with (6), $\mathbf{jt} \leqslant \mathbf{t}$. In other words, $\mathbf{t}\mathcal{R}$ is transitive for any relation $\mathcal{R}$. More generally, from $\mathbf{c}^{\cdot}, \mathbf{c}^{|}, \mathbf{i}, \mathbf{j} \leqslant \mathbf{t}$ and $\mathbf{tt} \leqslant \mathbf{t}$, we deduce that for any relation $\mathcal{R}$, $\mathbf{t}\mathcal{R}$ is a congruence containing both $\mathcal{R}$ and $\sim$.

In the context of proof assistants, this simple realisation makes it possible to reuse standard technology for automating equational reasoning (e.g., in the Coq proof assistant, setoid rewriting).

Also note that since $\sim\ =\ \mathbf{t}\perp$, we obtain as a special case that bisimilarity is a congruence. In particular, once the aforementioned technology has been settled for $\mathbf{t}\mathcal{R}$ for an arbitrary $\mathcal{R}$, tools for equational reasoning about bisimilarity come for free.

5. Streams

As a second example, we consider the *stream calculus*, as developed by Rutten [26]. Let us denote by $\mathbb{R}^{\omega}$ the set of *streams*, i.e., infinite sequences $\sigma, \tau \ldots$ of real numbers.

Together with the following function associating to each stream its first element and its tail, $\mathbb{R}^{\omega}$ is a final coalgebra for the functor $FX = \mathbb{R} \times X$

$$\mathbb{R}^{\omega} \to \mathbb{R} \times \mathbb{R}^{\omega}$$
$$\sigma \mapsto \langle \sigma_0, \sigma' \rangle$$

One can thus define streams by *behavioural differential equations* (i.e., F-coalgebras). For instance, the everywhere-0 stream $\widehat{0}$ can be defined by the following equations:

$$\widehat{0}_0 = 0 \qquad\qquad \widehat{0}' = \widehat{0}$$

Similarly, pointwise addition of streams can be defined by

$$(\sigma + \tau)_0 = \sigma_0 + \tau_0 \qquad (\sigma + \tau)' = \sigma' + \tau'$$

Shuffle product. Things become more interesting for more complex operations. Take for instance the *shuffle product* of streams, usually defined by the following formula:

$$(\sigma \otimes \tau)_n = \sum_{k=0}^{n} \binom{n}{k} \times \sigma_k \times \tau_{n-k}$$

This operation can alternatively be defined using the following differential equations, which no longer involve binomial coefficients:

$$(\sigma \otimes \tau)_0 = \sigma_0 \times \tau_0 \qquad (\sigma \otimes \tau)' = \sigma' \otimes \tau\ +\ \sigma \otimes \tau'$$

As noticed by Rutten, proving a simple property like associativity can be difficult with the former definition, as it would involve double summations of terms with several binomial coefficients. In contrast, one can give a straightforward coinductive proof.

Let $\mathbf{b}$ be the following (monotone) function on binary relations on streams:

$$\mathbf{b} :\ \mathcal{R} \mapsto \left\{ \langle \sigma,\ \tau \rangle \mid \sigma_0 = \tau_0 \text{ and } \sigma'\ \mathcal{R}\ \tau' \right\}$$

One easily checks that its greatest fixpoint is just the identity relation: $\langle \sigma, \tau \rangle \in \nu\mathbf{b}$ iff $\sigma = \tau$. One can thus prove stream equalities by coinduction.

As a trivial example consider commutativity of stream addition: it is immediate to see that the relation $\{ \langle \sigma + \tau,\ \tau + \sigma \rangle \mid \sigma, \tau \in \mathbb{R}^{\omega} \}$ is a post-fixpoint of $\mathbf{b}$; this relation is thus contained in the identity, and stream addition is commutative.

Coming back to associativity of the shuffle product, we might accordingly try to use the following relation:

$$\mathcal{S} \triangleq \{ \langle (\sigma \otimes \tau) \otimes \rho,\ \sigma \otimes (\tau \otimes \rho) \rangle \mid \sigma, \tau, \rho \in \mathbb{R}^{\omega} \}$$

Unfortunately, this relation is not a post-fixpoint of $\mathbf{b}$: assuming distributivity has already been proved, we have

$$((\sigma \otimes \tau) \otimes \rho)' = (\sigma' \otimes \tau) \otimes \rho\ +\ (\sigma \otimes \tau') \otimes \rho\ +\ (\sigma \otimes \tau) \otimes \rho'$$
$$(\sigma \otimes (\tau \otimes \rho))' = \sigma' \otimes (\tau \otimes \rho)\ +\ \sigma \otimes (\tau' \otimes \rho)\ +\ \sigma \otimes (\tau \otimes \rho')$$

and those two streams are not related by $\mathcal{S}$. Like in the previous example in CCS, we would like to cancel the two sums on both sides, in order to recover three pairs in $\mathcal{S}$. This is possible using the companion of $\mathbf{b}$: the following function is easily shown to be compatible for $\mathbf{b}$, so that $\mathcal{S} \leqslant \mathbf{b}^{\dagger}\mathcal{S}$.

$$\mathbf{c}^{+} : \mathcal{R} \mapsto \{ \langle \sigma + \rho,\ \tau + \omega \rangle \mid \sigma\ \mathcal{R}\ \tau,\ \rho\ \mathcal{R}\ \omega \}$$

We have thus obtained a straightforward proof of associativity of the shuffle product.

Exponentiation. Let us finally consider a third natural operation on streams: *exponentiation*, defined by the following differential equation:

$$e^{\sigma}_0 = e^{\sigma_0} \qquad\qquad e^{\sigma\prime} = \sigma' \otimes e^{\sigma}$$

As expected, we have $e^{\sigma+\tau} = e^{\sigma} \otimes e^{\tau}$. To prove it by following the same path as above, one is leaded to cancel a shuffle product, thus calling for the following function:

$$\mathbf{c}^{\otimes} : \mathcal{R} \mapsto \{ \langle \sigma \otimes \rho,\ \tau \otimes \omega \rangle \mid \sigma\ \mathcal{R}\ \tau,\ \rho\ \mathcal{R}\ \omega \}$$

While this function is indeed below the companion of $\mathbf{b}$, it is not compatible for $\mathbf{b}$.

To understand why, let us try to prove compatibility of this function, i.e., $\mathbf{c}^{\otimes}\mathbf{b} \leqslant \mathbf{bc}^{\otimes}$. Let $\mathcal{R}$ be a relation. We have

$$\mathbf{c}^{\otimes}\mathbf{b}\mathcal{R} = \{ \langle \sigma \otimes \rho,\ \tau \otimes \omega \rangle \mid \sigma\ \mathbf{b}\mathcal{R}\ \tau,\ \rho\ \mathbf{b}\mathcal{R}\ \omega \}$$

So assuming $\sigma\ \ \mathbf{b}\mathcal{R}\ \ \tau$ and $\rho\ \ \mathbf{b}\mathcal{R}\ \ \omega$, we have to show that $\langle \sigma \otimes \rho,\ \tau \otimes \omega \rangle \in \mathbf{bc}^{\otimes}\mathcal{R}$. That $(\sigma \otimes \rho)_0 = (\tau \otimes \omega)_0$ is easy; the problem comes from the tails of those streams:

$$(\sigma \otimes \rho)' = \sigma' \otimes \rho\ +\ \sigma \otimes \rho'$$
$$(\tau \otimes \omega)' = \tau' \otimes \omega\ +\ \tau \otimes \omega'$$

First we need to cancel the sum operation (using $\mathbf{c}^+$). Second, while we have $\sigma' \mathcal{R} \tau'$ and $\rho' \mathcal{R} \omega'$ by assumption, we only have $\rho \, \mathbf{b}\mathcal{R} \, \omega$ and $\sigma \, \mathbf{b}\mathcal{R} \, \tau$. In the end, instead of $\mathbf{c}^\otimes \mathbf{b} \leqslant \mathbf{b}\mathbf{c}^\otimes$, we have

$$\mathbf{c}^\otimes \mathbf{b} \leqslant \mathbf{b}\mathbf{c}^+\mathbf{c}^\otimes(\mathbf{b} \vee 1) \tag{12}$$

We shall see in the following section that such a result nevertheless ensures that the function $\mathbf{c}^\otimes$ is below the companion of $\mathbf{b}$, and can thus safely be used in enhanced coinductive proofs about streams.

6. Compatibility up-to

In this section we show that the companion is a coinductive object. This gives us powerful proof techniques to obtain enhancements.

Definition 6.1. Let B be the following function from monotone functions on X to monotone functions on X:

$$\begin{aligned} B : [X \to X] &\to [X \to X] \\ g &\mapsto \bigvee_{fb \leqslant bg} f \end{aligned}$$

Lemma 6.2. *B is monotone, and for all functions $f, g : [X \to X]$,*

$$f \leqslant Bg \quad \textit{iff} \quad fb \leqslant bg \tag{13}$$

In particular, f is compatible if and only if it is a post-fixpoint of B, so that the companion is the greatest fixpoint of B:

$$t = \nu B \tag{14}$$

We can thus reuse the machinery from Section 3 with B, in the second-order lattice $[X \to X]$. Let T the companion of B, and let $B^\dagger$ be the corresponding enhancement of B: $B^\dagger \triangleq BT$.

In the previous section, in the example about streams, we had a first function $\mathbf{c}^+$, which was compatible and thus trivially below the companion. In contrast, we claimed that the function $\mathbf{c}^\otimes$ was below the companion, although it is not compatible stricto senso: we do not have $\mathbf{c}^\otimes \mathbf{b} \leqslant \mathbf{b}\mathbf{c}^\otimes$, i.e., $\mathbf{c}^\otimes \leqslant \mathbf{B}\mathbf{c}^\otimes$ (where $\mathbf{B}$ is the higher-order function associated to the function $\mathbf{b}$ from Section 5). Instead, we had $\mathbf{c}^\otimes \leqslant \mathbf{B}(\mathbf{c}^+\mathbf{c}^\otimes(\mathbf{b} \vee 1))$. Using the results below, we will deduce from this inequality that $\mathbf{c}^\otimes \leqslant \mathbf{B}^\dagger \mathbf{c}^\otimes$, so that $\mathbf{c}^\otimes \leqslant \nu\mathbf{B}$: the function $\mathbf{c}^\otimes$ indeed lives below the companion.

In a sense, we face the standard scenario of bisimulation proofs, but in the lattice of monotone functions: $\mathbf{c}^\otimes$ is an obvious coinductive candidate, but it is too weak, we should strengthen it to get a post-fixpoint. Luckily, instead of doing so, we can use an enhancement to build on the knowledge accumulated so far about the companion (in this case, that it contains $\mathbf{c}^+$, amongst other things.)

Getting back to the abstract framework, the second-order companion T enjoys many good properties, listed in Proposition 6.4 below. We first need to establish a compatibility result for B.

Lemma 6.3. *The function $S : f \mapsto ff$ is compatible for B.*

Proof. We have

$$\begin{aligned} & SB \leqslant BS \\ \text{iff} \quad & \forall f, (Bf)(Bf) \leqslant B(ff) && \text{(by definition of } S\text{)} \\ \text{iff} \quad & \forall f, (Bf)(Bf)b \leqslant bff && \text{(by (13))} \end{aligned}$$

Fix some monotone function $f : [X \to X]$. By taking $g = f$ and $f = Bf$ in (13) we have $(Bf)b \leqslant bf$. Thus we get

$$(Bf)(Bf)b \leqslant (Bf)bf \leqslant bff$$

and S is compatible for B. □

Proposition 6.4. *For any function $f : [X \to X]$, we have*

$$t \leqslant Tf \tag{15}$$
$$b \leqslant Tf \tag{16}$$
$$1 \leqslant Tf \tag{17}$$
$$f \leqslant Tf \tag{18}$$
$$TTf \leqslant Tf \tag{19}$$
$$(Tf)(Tf) \leqslant Tf \tag{20}$$

In particular, Tf is always an idempotent function.

Proof.
- (15): recall that $t = T\bot$, and T is monotone;
- (16),(17): trivial consequences of the previous point and (4) and (5), respectively;
- (18),(19): respective instances of (5) and (6) for B (thus moving to the lattice $[X \to X]$);
- (20): by Lemma 6.3 we get $S \leqslant T$. Then it suffices to compute: $(Tf)(Tf) = STf \leqslant TTf \leqslant Tf$. □

Note that the way we obtain (20) is very similar to the way we establish transitivity of $\mathbf{t}\mathcal{R}$ at the end of Section 4.

Coming back to the example about the shuffle product on streams, write $\mathbf{t}$ and $\mathbf{T}$ for the companions of $\mathbf{b}$ and $\mathbf{B}$. It is now straightforward to check that $\mathbf{c}^+\mathbf{c}^\otimes(\mathbf{b} \vee 1) \leqslant \mathbf{T}\mathbf{c}^\otimes$:

$$\begin{aligned} \mathbf{c}^+\mathbf{c}^\otimes(\mathbf{b} \vee 1) &\leqslant \mathbf{t}\mathbf{c}^\otimes(\mathbf{b} \vee 1) && (\mathbf{c}^+ \text{ is compatible}) \\ &\leqslant (\mathbf{T}\mathbf{c}^\otimes)\mathbf{c}^\otimes(\mathbf{T}\mathbf{c}^\otimes \vee \mathbf{T}\mathbf{c}^\otimes) \\ & && \text{(by (15), (16), and (17))} \\ &\leqslant (\mathbf{T}\mathbf{c}^\otimes)(\mathbf{T}\mathbf{c}^\otimes)(\mathbf{T}\mathbf{c}^\otimes) && \text{(by (18))} \\ &\leqslant \mathbf{T}\mathbf{c}^\otimes && \text{(by (20) twice)} \end{aligned}$$

So from (12) we deduce $\mathbf{c}^\otimes \leqslant \mathbf{B}^\dagger \mathbf{c}^\otimes$ and thus $\mathbf{c}^\otimes \leqslant \mathbf{t}$, as announced earlier.

(Note that we chose to make the function $\mathbf{c}^+\mathbf{c}^\otimes(\mathbf{b} \vee 1)$ explicit here for the sake of explanation. In a direct proof, one would prove $\mathbf{c}^\otimes \leqslant \mathbf{B}^\dagger \mathbf{c}^\otimes$ by extracting the required components out of $\mathbf{T}\mathbf{c}^\otimes$ on the fly, exactly as we did in Section 4 but at the second-order level.)

7. Symmetry arguments

We now give a rather general result allowing to exploit symmetry arguments in various proofs. This result formally justifies standard practice in bisimulation proofs with paper and pencil. When it comes to formal, mechanised, proofs, it is crucial to have such results, to factor the code and avoid cut-and-paste.

Let $i : [X \to X]$ be a monotone involution on X:

$$ii = 1 \tag{21}$$

Call an element $x \in X$ *symmetric* if $ix = x$ (which is equivalent to $ix \leqslant x$ thanks to (21)). Call a function $f : [X \to X]$ *symmetric* if $fi = if$ (which is equivalent to f being compatible for i, again using (21)).

As a concrete example in the lattice of binary relations, the natural candidate for the function i is the transposition function from Section 4:

$$\mathbf{i} : \mathcal{R} \mapsto \mathcal{R}^{-1} = \{\langle Q, \, P\rangle \mid P \, \mathcal{R} \, Q\}$$

With such a choice, a relation $\mathcal{R}$ is symmetric if $\mathcal{R}^{-1} = \mathcal{R}$, and a function f is symmetric if $f(\mathcal{R}^{-1}) = f(\mathcal{R})^{-1}$ for all relation $\mathcal{R}$.

When the function b is symmetric, the following proposition can be used factor out proofs about symmetric candidates, both at the level of elements (X) and at the level of enhancements ($[X \to X]$). We instantiate this result in the following section, when reasoning about bisimilarity in CCS and the π-calculus.

$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'}\alpha \neq a, \overline{a} \qquad \frac{!P|P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'}$$

Figure 2. Remaining rules for the LTS of CCS.

Proposition 7.1. *Let $s : [X \to X]$ be a monotone function such that the function b decomposes as follows:*

$$b = s \wedge isi$$

Then νb is symmetric, $it = t$, and

1. *for all $x, y \in X$ with x symmetric,*
$$x \leqslant bty \qquad \textit{iff} \qquad x \leqslant sty \tag{22}$$
2. *for all $f, g : [X \to X]$ with f symmetric,*
$$fb \leqslant bTg \qquad \textit{iff} \qquad fb \leqslant sTg \tag{23}$$

Proof. First of all, i is compatible for b:

$$ib = i(s \wedge isi) \leqslant is \wedge iisi = isii \wedge si = (isi \wedge s)i = bi$$

Whence $i \leqslant t$. With (6) and (21), it follows easily that $it = t$, of which $i\nu b = \nu b$ is a special case since $\nu b = t\bot$. In the subsequent equivalences, the forward implications follow from $b \leqslant s$. For the converse implications:

1. assume that $x \leqslant sty$ with x symmetric. We have
$$x = ix \leqslant isty = isity$$
and thus $x \leqslant sty \wedge isity = bty$.
2. first note that $(Tg)i = i(Tg)$: we have
$$i(Tg) \leqslant t(Tg) \leqslant (Tg)(Tg) \leqslant Tg$$
whence $iTg = Tg$ by using (21); we get $(Tg)i = Tg$ by a similar argument.
Then assume that $fb \leqslant sTg$ with f symmetric. We have
$$fb = fiib = ifbi \leqslant is(Tg)i = isiTg$$
and thus $fb \leqslant sTg \wedge isiTg = bTg$. □

8. Examples: up-to congruence for CCS and pi

In this section, we illustrate the above framework by applying it to recover up-to-context and up-to congruence for CCS and the π-calculus, in compositional way. Indeed, thanks to the closure properties of the companion, it suffices to show that the functions associated to each syntactic construction are below $\mathbf{t}$ to obtain the full context closure function. Combined with the fact that the transposition function $\mathbf{i}$ and the squaring function $\mathbf{j}$ are also below $\mathbf{t}$, we will immediately obtain the full congruence closure.

8.1 CCS

For the sake of completeness, let us consider here the entire calculus of communicating systems [19]. In addition to the operations used in Section 4, there is choice, name restriction, and replication.

$$P, Q ::= A \mid 0 \mid \alpha.P \mid P|P \mid P + P \mid (\nu a)P \mid !P$$
$$\alpha, \beta ::= a \mid \overline{a} \mid \tau$$

The additional rules for the labelled transition system are given in Figure 2. (The symmetrical rule for choice is omitted.)

Recall the function $\mathbf{b}$ from Section 4, which we used to define bisimilarity. Let $\mathbf{s}$ be the "first half" of this function:

$$\mathbf{s} : \mathcal{R} \mapsto \{\langle P, Q\rangle \mid \forall \alpha, P', P \xrightarrow{\alpha} P' \text{ entails } \exists Q', Q \xrightarrow{\alpha} Q' \text{ and } P' \mathcal{R} Q'\}$$

With such a function, we only play from left to right. The post-fixpoints of $\mathbf{s}$ are the *simulations*, and its greatest fixpoint is *similarity*. The composite function $\mathbf{isi}$ corresponds to simulations again, but played from right to left. As expected the function $\mathbf{b}$ for bisimilarity thus decomposes as required in Proposition 7.1:

$$\mathbf{b} = \mathbf{s} \wedge \mathbf{isi} \tag{24}$$

Applied in this setting, equivalence (22) is not so surprising: when analysing the transitions of a symmetric bisimulation candidate, we can restrict ourselves to the left-to-right part of the bisimulation game. Note that thanks to the companion, we do not need y to be symmetric (because ty is). The second equivalence (23) is quite important in the sequel: one can also restrict ourselves to the left-to-right part of the bisimulation game when analysing the behaviour of a potential enhancement, provided it is symmetric.

Following closely the syntax of the calculus, we define the following functions on binary relations:

$$\mathbf{c}^{\cdot} : \mathcal{R} \mapsto \{\langle \alpha.P, \alpha.Q\rangle \mid \alpha \text{ a label}, P \mathcal{R} Q\}$$
$$\mathbf{c}^{|} : \mathcal{R} \mapsto \{\langle P|P', Q|Q'\rangle \mid P \mathcal{R} Q,\ P' \mathcal{R} Q'\}$$
$$\mathbf{c}^{+} : \mathcal{R} \mapsto \{\langle P + P', Q + Q'\rangle \mid P \mathcal{R} Q,\ P' \mathcal{R} Q'\}$$
$$\mathbf{c}^{!} : \mathcal{R} \mapsto \{\langle !P, !Q\rangle \mid P \mathcal{R} Q\}$$
$$\mathbf{c}^{\nu} : \mathcal{R} \mapsto \{\langle (\nu a)P, (\nu a)Q\rangle \mid \alpha \text{ a name}, P \mathcal{R} Q\}$$

The functions $\mathbf{c}^{|}$ and $\mathbf{c}^{\cdot}$ have already been defined in Section 4; we include them here to emphasise the uniformity of those definitions.

Let c be one of the above functions; we want to prove $c \leqslant \mathbf{t}$, where $\mathbf{t}$ is the companion of $\mathbf{b}$. From the results of Section 6, it thus suffices to prove $c\mathbf{b} \leqslant \mathbf{b}\mathbf{T}c$, where $\mathbf{T}$ is the companion of the second-order function $\mathbf{B}$ associated to $\mathbf{b}$. And since all the above functions are symmetric, it suffices by (23) to prove

$$c\mathbf{b} \leqslant \mathbf{s}\mathbf{T}c$$

For the "dynamic" operations that disappear after a single transition (functions $\mathbf{c}^{\cdot}$ and $\mathbf{c}^{+}$), we actually do not need coinduction at all. Routine computations lead to

$$\mathbf{c}^{\cdot}\mathbf{b} \leqslant \mathbf{s}\mathbf{b} \qquad \mathbf{c}^{+}\mathbf{b} \leqslant \mathbf{s}$$

(For $\mathbf{c}^{\cdot}$, we have $\mathbf{c}^{\cdot} \leqslant \mathbf{s}$.) This is fine because $\mathbf{b}, 1 \leqslant \mathbf{T}\bot$ (16, 17).

Instead, we do need coinduction for the "static" operations, which persist through transitions. The simplest is name restriction: we have $\mathbf{c}^{\nu}\mathbf{b} \leqslant \mathbf{s}\mathbf{c}^{\nu}$, and $\mathbf{c}^{\nu} \leqslant \mathbf{T}\mathbf{c}^{\nu}$ by (18). Parallel composition and replication require more care, we give detailed proofs to better illustrate our method.

Lemma 8.1. *We have $\mathbf{c}^{|}\mathbf{b} \leqslant \mathbf{s}\mathbf{T}\mathbf{c}^{|}$, whence $\mathbf{c}^{|} \leqslant \mathbf{t}$.*

Proof. Let $\mathcal{R}$ be a relation, an let P, R, Q, S be processes such that $\langle P, Q\rangle, \langle R, S\rangle \in \mathbf{b}\mathcal{R}$. We have to show that $\langle P|R, Q|S\rangle$ belongs to $\mathbf{s}\mathbf{T}\mathbf{c}^{|}$. Thus suppose that $P|R \xrightarrow{\alpha} P_0$ and let us find some Q_0 such that $Q|S \xrightarrow{\alpha} Q_0$ and $\langle P_0, Q_0\rangle \in \mathbf{T}\mathbf{c}^{|}\mathcal{R}$. There are four cases according to the rules of parallel composition (Figure 1):

1. $P_0 = P'|R'$ with $\alpha = \tau$, and for some name a, $P \xrightarrow{a} P'$, and $R \xrightarrow{\overline{a}} R'$. From our assumptions about $\langle P, Q\rangle$ and $\langle R, S\rangle$, we obtain processes Q' and S' such that $Q \xrightarrow{a} Q'$, $S \xrightarrow{\overline{a}} S'$, $P' \mathcal{R} Q'$ and $R' \mathcal{R} S'$. We deduce that $Q|S \xrightarrow{\tau} Q'|S'$, and the pair $\langle P'|R', Q'|S'\rangle$ belongs to $\mathbf{c}^{|}\mathcal{R}$ and hence $\mathbf{T}\mathbf{c}^{|}\mathcal{R}$ by (18).
2. same as above but with a and $\overline{a}$ exchanged.
3. $P_0 = P'|R$ with $P \xrightarrow{\alpha} P'$. From the hypothesis about the pair $\langle P, Q\rangle$ we obtain a process Q' such that $Q \xrightarrow{a} Q'$ and $P' \mathcal{R} Q'$. We deduce that $Q|S \xrightarrow{\alpha} Q'|S$, and it remains to show
$$\langle P'|R, Q'|S\rangle \in \mathbf{T}\mathbf{c}^{|}\mathcal{R}$$

This is not as direct as before: rather than $R\ \mathcal{R}\ S$, we have $R\ \mathbf{b}\mathcal{R}\ S$. From (18) and (20), we have $\mathbf{c}^{!}\mathbf{T}\mathbf{c}^{!} \leqslant \mathbf{T}\mathbf{c}^{!}$. Therefore, it suffices to show that $P'\ \mathbf{T}\mathbf{c}^{!}\mathcal{R}\ Q'$ and $R\ \mathbf{T}\mathbf{c}^{!}\mathcal{R}\ S$. The former holds thanks to (17); for the latter we use (16) instead.

4. $P_0 = P|R'$ with $R \xrightarrow{\alpha} R'$. This case is handled as above. □

(Note that the above proof amounts to proving $\mathbf{c}^{!}\mathbf{b} \leqslant \mathbf{s}\mathbf{c}^{!}(\mathbf{b}\vee 1)$ and then showing that $\mathbf{c}^{!}(\mathbf{b}\vee 1) \leqslant \mathbf{T}\mathbf{c}^{!}$ using the generic properties of $\mathbf{T}$—Proposition 6.4.)

Finally consider replication. This operation is quite challenging as far as up-to techniques are concerned: there were a slight mistake in [28], it requires specific rule formats [25], and people formalising up-to techniques in proof assistants have eluded this operation so far [7, 21]. The techniques developed therein allow us to give a much cleaner proof, thus amenable to formalisation.

Let us first discuss the LTS rule for replication.

- The one presented in Figure 2 is standard; it makes it trivial to prove $!P \sim !P|P$ (which is the intended meaning) but it has the drawback of making the LTS infinitary branching.
- The obvious rule "$P \xrightarrow{\alpha} P'$ entails $!P \xrightarrow{\alpha} !P|P$" is not enough in presence of choice, as it does not allow $!(a.0 + \overline{a}.0)$ to perform internal transitions.
- In his book about the π-calculus [28], Sangiorgi prefers to inline the behaviour of parallel composition to keep the LTS finitely branching. But this leads to duplication: there are two rules for replication in CCS, and three in the π-calculus.

Here we propose a path which is agnostic about this choice: it relies only on the following proposition, which is always valid in CCS and in π, whatever their presentation.

Proposition 8.2.

(i) If $!P \xrightarrow{\alpha} P_0$, then there exists P_1 such that $P|P \xrightarrow{\alpha} P_1$ and $P_0 \sim !P|P_1$.

(ii) Conversely, if $P|P \xrightarrow{\alpha} P_1$, then there exists P_0 such that $!P \xrightarrow{\alpha} P_0$ and $P_0 \sim !P|P_1$.

We need another preliminary result:

Lemma 8.3. *The following function is compatible.*

$$\mathbf{k} : \mathcal{R} \mapsto \{\langle P, Q\rangle \mid \exists P'Q',\ P \sim P',\ P'\ \mathcal{R}\ Q',\ Q' \sim Q\}$$

Lemma 8.4. *We have $\mathbf{c}^{!}\mathbf{b} \leqslant \mathbf{s}\mathbf{T}\mathbf{c}^{!}$, whence $\mathbf{c}^{!} \leqslant \mathbf{t}$.*

Proof. Let $\mathcal{R}$ be a relation, let $\langle P, Q\rangle \in \mathbf{b}\mathcal{R}$. We have to show that $\langle !P, !Q\rangle$ belongs to $\mathbf{s}\mathbf{T}\mathbf{c}^{!}$. Thus suppose that $!P \xrightarrow{\alpha} P_0$ and let us find some Q_0 such that $!Q \xrightarrow{\alpha} Q_0$ and $\langle P_0, Q_0\rangle \in \mathbf{T}\mathbf{c}^{!}\mathcal{R}$.

By Proposition 8.2(i), there is some P_1 such that $P|P \xrightarrow{\alpha} P_1$ and $P_0 \sim !P|P_1$. Now notice that

$$\begin{aligned}
\langle P|P, Q|Q\rangle &\in \mathbf{c}^{!}\mathbf{b}\mathcal{R} && \text{(by definition of } \mathbf{c}^{!}\text{)}\\
&\leqslant \mathbf{t}\mathbf{b}\mathcal{R} && \text{(by Lemma 8.1)}\\
&\leqslant \mathbf{b}\mathbf{t}\mathcal{R} && \text{(the companion is compatible (3))}
\end{aligned}$$

By definition of $\mathbf{b}$, we thus obtain a process Q_1 such that $Q|Q \xrightarrow{\alpha} Q_1$ and $P_1\ \mathbf{t}\mathcal{R}\ Q_1$. We continue with Proposition 8.2(ii) which gives us Q_0 such that $!Q \xrightarrow{\alpha} Q_0$ and $Q_0 \sim !Q|Q_1$.

Summarising our assumptions, we have

$$!P \xrightarrow{\alpha} P_0 \ \sim\ !P|P_1 \qquad \begin{matrix} P\ \mathbf{b}\mathcal{R}\ Q \\ P_1\ \mathbf{t}\mathcal{R}\ Q_1 \end{matrix} \qquad !Q|Q_1 \ \sim\ Q_0 \xleftarrow{\alpha} !Q$$

It remains to show that $\langle P_0, Q_0\rangle \in \mathbf{T}\mathbf{c}^{!}\mathcal{R}$. This follows from the closure properties of $\mathbf{T}$ (Proposition 6.4), and $\mathbf{c}^{!}, \mathbf{k} \leqslant \mathbf{t}$ (Lemmas 8.1 and 8.3). □

$$\frac{}{\overline{a}b.P \xrightarrow{\overline{a}b} P} \qquad \frac{}{a(b).P \xrightarrow{a(c)} P\{^{c}/_{b}\}} \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\mathrm{bn}(\alpha)\#Q$$

$$\frac{P \xrightarrow{a(b)} P' \qquad Q \xrightarrow{\overline{a}b} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \frac{P \xrightarrow{a(b)} P' \qquad Q \xrightarrow{\overline{a}\nu b} Q'}{P|Q \xrightarrow{\tau} (\nu b)(P'|Q')}b\#P'$$

$$\frac{P \xrightarrow{\alpha} P'}{(\nu b)P \xrightarrow{\alpha} (\nu b)P'}b\#\mathrm{n}(\alpha) \qquad \frac{P \xrightarrow{\overline{a}b} P'}{(\nu b)P \xrightarrow{\overline{a}\nu b} P'}a \neq b$$

Figure 3. LTS rules for prefix, parallel composition, and name restriction in the π-calculus.

8.2 The π-calculus

The π-calculus [20] differs from CCS in that names can be passed along synchronisations. We briefly recall the main definitions here, referring to Sangiorgi's book for a more detailed exposition of this calculus [28]. The syntax of processes remains the same as in CCS, only the notion of prefix changes:

$$\begin{aligned}
P, Q &::= A \mid 0 \mid \pi.P \mid P|P \mid P + P \mid (\nu a)P \mid !P\\
\pi &::= a(b) \mid \overline{a}b && \text{(prefixes)}\\
\alpha &::= \pi \mid \overline{a}\nu b \mid \tau && \text{(labels)}
\end{aligned}$$

The prefix $a(b)$ denotes the input on a of a name b, bound in the continuation; $\overline{a}b$ denotes the output on a of a name b. Labels include two additional constructs: τ for internal communications, and $\overline{a}\nu b$, for the emission of a private name b, bound in the continuation.

Concerning the (early) LTS, the rules for process constants, choice, and replication are the same as in CCS, the other ones are given in Figure 3. The three symmetrical rules for parallel composition are omitted. Processes are considered modulo alpha equivalence. The functions $\mathrm{n}(\cdot)$ (names of a label) and $\mathrm{bn}(\cdot)$ (bound names of a label) are defined as follows:

α	$a(b)$	$\overline{a}b$	$\overline{a}\nu b$	τ
$\mathrm{n}(\pi)$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\emptyset$
$\mathrm{bn}(\pi)$	$\emptyset$	$\emptyset$	$\{b\}$	$\emptyset$

To define (early) bisimilarity, one can import the function $\mathbf{b}$ from CCS almost as is. The only peculiarity is that we should ignore bound output transitions $\overline{a}\nu b$ when b occurs free in the answering process. Indeed, one should equate the following processes

$$P \triangleq (\nu b)\overline{a}b.0 \qquad\qquad Q \triangleq P|(\nu c)\overline{c}d.0$$

(Because $(\nu c)\overline{c}d.0$ is clearly equivalent to 0.) However, while P can perform a transition labelled $\overline{a}\nu b$ for any $b \neq a$, the process Q can only perform such a transition when $b \neq a, d$, because of its free (yet useless) occurrence of d.

Whence the appropriate definitions of functions $\mathbf{s}$ and $\mathbf{b}$:

$$\begin{aligned}
\mathbf{s} :\ \mathcal{R} \mapsto\ &\{\langle P, Q\rangle \mid \forall \alpha,\ \mathrm{bn}(\alpha)\#Q \text{ and } P \xrightarrow{\alpha} P'\\
&\qquad \text{entail } \exists Q',\ Q \xrightarrow{\alpha} Q' \text{ and } P'\ \mathcal{R}\ Q'\}
\end{aligned}$$

$$\mathbf{b} \triangleq \mathbf{s} \wedge \mathbf{i}\mathbf{s}\mathbf{i}$$

The functions $\mathbf{c}^{!}, \mathbf{c}^{+}, \mathbf{c}^{\nu}, \mathbf{c}^{!}$ are defined exactly as in CCS; we however replace the function $\mathbf{c}^{\cdot}$ with the two following ones:

$$\mathbf{c}^{o} : \mathcal{R} \mapsto \{\langle \overline{a}b.P, \overline{a}b.Q\rangle \mid a, b \text{ names}, P\ \mathcal{R}\ Q\}$$

$$\mathbf{c}^{i} : \mathcal{R} \mapsto \{\langle a(b).P, a(b).Q\rangle \mid a, b \text{ names}, \forall c, P\{^{c}/_{b}\}\ \mathcal{R}\ Q\{^{c}/_{b}\}\}$$

The reason why we need such a quantification on all names c in the definition of $\mathbf{c}^{i}$ is that bisimilarity is not preserved by input prefixes in the π-calculus: it is not closed under substitution, and the input prefix brings such substitutions. Thus, the plain function

corresponding to input prefix cannot be below the companion (it always preserves bisimilarity (8)). Instead, the above definition of $\mathbf{c}^i$ takes into account the fact that b might be substituted later on.

Like in CCS, it is routine to check that we have

$$\mathbf{c}^o\mathbf{b} \leqslant \mathbf{sb} \qquad \mathbf{c}^i\mathbf{b} \leqslant \mathbf{sb} \qquad \mathbf{c}^+\mathbf{b} \leqslant \mathbf{s}$$

Whence $c \leqslant \mathbf{t}$ for $c \in \{\mathbf{c}^o, \mathbf{c}^i, \mathbf{c}^+\}$. Name restriction is no longer compatible stricto-senso: we have

$$\mathbf{c}^\nu\mathbf{b} \leqslant \mathbf{s}(\mathbf{c}^\nu \vee 1)$$

The identity comes from the opening rule, where name restriction disappears. This is however sufficient to conclude that $\mathbf{c}^\nu \leqslant \mathbf{t}$.

The proof for parallel composition also has to be slightly adapted, to take care of the new side conditions in the rules, but also because of scope extrusion: in the second synchronisation rule, a name restriction appears, which was not there in CCS. Accordingly, one can prove

$$\mathbf{c}^|\mathbf{b} \leqslant \mathbf{s}(\mathbf{c}^|(\mathbf{b} \vee 1) \vee \mathbf{c}^\nu\mathbf{c}^|)$$

Since we already obtained $\mathbf{c}^\nu \leqslant \mathbf{t}$, this suffices to deduce $\mathbf{c}^| \leqslant \mathbf{t}$.

Replication is handled exactly as in the previous section (our proof of Lemma 8.4 is generic). The counterpart of Lemma 8.3 however requires more care, because of the condition on bound output transitions in the definition of $\mathbf{s}$. The problem is well-known: although bisimilarity is transitive in the π-calculus, the composition of two bisimulations is not always a bisimulation [28]. The reason appears explicitly in the following proof.

Lemma 8.5 (Lemma 8.3 for π)**.** *In the π-calculus, the following functions are below* $\mathbf{t}$

$$\boldsymbol{\sigma} : \mathcal{R} \mapsto \{\langle \sigma P, \sigma Q\rangle \mid \sigma \textit{ injective name substitution}, P\ \mathcal{R}\ Q\}$$
$$\mathbf{j} : \mathcal{R} \mapsto \{\langle P, R\rangle \mid \exists Q,\ P\ \mathcal{R}\ Q,\ Q\ \mathcal{R}\ R\}$$
$$\mathbf{k} : \mathcal{R} \mapsto \{\langle P, Q\rangle \mid \exists P'Q',\ P \sim P',\ P'\ \mathcal{R}\ Q',\ Q' \sim Q\}$$

Proof. The function $\boldsymbol{\sigma}$ is actually compatible. In contrast, the squaring function $\mathbf{j}$ is not compatible in π. We use second-order enhancements and symmetry arguments to reduce the problem to $\mathbf{jb} \leqslant \mathbf{sTj}$. To prove this inclusion, let $\mathcal{R}$ be a relation and assume $P\ \mathbf{b}\mathcal{R}\ Q$ and $Q\ \mathbf{b}\mathcal{R}\ R$ for some processes P, Q, R. We have to show that $\langle P, R\rangle \in \mathbf{sTj}$. Further assume $P \xrightarrow{\alpha} P'$.

- If α is not a bound output, then $\mathrm{bn}(\alpha) = \emptyset$ and the proof is standard: we first get Q' such that $Q \xrightarrow{\alpha} Q'$ and $P'\ \mathcal{R}\ Q'$, and then R' such that $R \xrightarrow{\alpha} R'$ and $Q'\ \mathcal{R}\ R'$. We finally check that $P'\ \mathbf{j}\mathcal{R}\ R'$, and thus $P'\ \mathbf{Tj}\mathcal{R}\ R'$.
- If instead $\alpha = \overline{a}\nu b$, then we know by assumption that $b\#R$, but we would need $b\#Q$ to perform the first step above. Since this might not be the case, we pick some name c fresh for both P, Q, and R, and we use the permutation $\sigma = (bc)$ to play a challenge on $\overline{a}\nu c$ instead: we have $P \xrightarrow{\overline{a}\nu c} \sigma P'$ with $c\#Q$, so that we get Q' such that $Q \xrightarrow{\overline{a}\nu c} Q'$ and $\sigma P'\ \mathcal{R}\ Q'$. Then since $c\#R$, we get R' such that $R \xrightarrow{\overline{a}\nu c} R'$ and $Q'\ \mathcal{R}\ R'$. Since $b, c\#R$, it follows that $R \xrightarrow{\overline{a}\nu b} \sigma R'$. We have $\sigma P'\ \mathbf{j}\mathcal{R}\ \sigma R'$. Since σ is injective, we get $\sigma\sigma P'\ \boldsymbol{\sigma}\mathbf{j}\mathcal{R}\ \sigma\sigma R'$. From $\sigma\sigma = 1$ and $\boldsymbol{\sigma} \leqslant \mathbf{t}$, we finally obtain $P'\ \mathbf{Tj}\mathcal{R}\ R'$.

That $\mathbf{k} \leqslant \mathbf{t}$ follows from $\mathbf{j} \leqslant \mathbf{t}$ and the closure properties of $\mathbf{t}$ (Lemma 3.2). □

9. Respectful vs. compatible

Before turning to parametric coinduction, we discuss an historical peculiarity which caused some troubles since the introduction of up-to techniques, and which we can nicely solve by using the companion function.

When Sangiorgi studied the bisimulation proof technique [27], he introduced the notion of *respectful* function to obtain compositionality results. With the present notation, a monotone function $f : [X \to X]$ is respectful (for b) if for all $x, y \in X$,

$$x \leqslant y \text{ and } x \leqslant by \text{ entail } fx \leqslant bfy\ .$$

Let $b' \triangleq b \wedge 1$ (i.e., $b'y = by \wedge y$). Without the assumption $x \leqslant y$, respectfulness would be equivalent to compatibility (for b). With this assumption, it is equivalent to compatibility for b'.

One can easily show that any compatible function (for b) is respectful, but some interesting respectful functions are not compatible. This is the reason why Sangiorgi needed this refinement. For instance the context closure function in CCS is respectful but not compatible. Hur et al. used respectfulness for the same reason [11].

In our own previous work [18, 22, 24], some of which with Sangiorgi, we found that the theory of plain compatibility was somewhat nicer to develop than that of respectfulness, so that we proposed to use the function b' rather than b when necessary (doing so is always possible). Although this is not the only reason, Pohjola and Parrow also chose a function b such that $b = b'$ in their theory of up-to techniques for the psi-calculus [21].

In this paper, we used the most natural function $\mathbf{b}$ to define strong bisimilarity in CCS, and this function does not satisfies $\mathbf{b} = \mathbf{b}'$. So how is it possible that we could obtain up-to context?

The point is that with the companion function, we do not need the up-to context function to be compatible stricto-senso. It just has to be below the companion function t. For instance, in our proof for the parallel composition operation in CCS, we obtained $\mathbf{c}^|\mathbf{b} \leqslant \mathbf{bTc}^|$, which does not entail compatibility of $\mathbf{c}^|$.This contrasts with the literature, where we would prove $\mathbf{c}^|\mathbf{b}' \leqslant \mathbf{b}'\mathbf{c}^|$, i.e., that $\mathbf{c}^|$ is respectful (compatible for $\mathbf{b}'$).

That we can recover up-to context in CCS and π without switching to $\mathbf{b}'$ is not a coincidence: the greatest respectful function always coincides with the greatest compatible function:

Proposition 9.1. *Let t' be the companion of b'. We have $t' = t$ and $b'^\dagger = b^\dagger$*

Proof. Any function compatible for b is compatible for b', whence $t \leqslant t'$ $(*)$. Then we show $b \leqslant b't$ $(**)$. We have $b't = bt \wedge t$; we get $b \leqslant bt$ using (5), and $b \leqslant t$ is just (4).

To obtain $t' \leqslant t$, we show that t' is compatible for b:

$$\begin{aligned} t'b &\leqslant t'b't && \text{(by } (**)\text{)} \\ &\leqslant b't't && \text{(compatibility of } t'\text{)} \\ &\leqslant b't' && \text{(by } (*) \text{ and idempotence of } t'\text{)} \\ &\leqslant bt' && (b' \leqslant b') \end{aligned}$$

Finally, $bt \leqslant b'tt = b't \leqslant bt$ by $(**)$, idempotence of t, and the fact that $b' \leqslant b$. So that $b'^\dagger = b't' = b't = bt = b^\dagger$. □

In other words, the historical tradeoff between b and b', or compatibility and respectfulness, is irrelevant. The functions b and b' lead to the same coinductive proof principle once enhanced with their companion. One can actually go even further and show that obtaining specific up-to techniques is equally hard with b' and b: their (enhanced) second-order proof principles also collapse.

Proposition 9.2. *Let B' be the second-order function associated to b', and T' be the companion of B' (so that $\nu B' = t' = T'\bot$). We have $T' = T$ and $B'^\dagger = B^\dagger$.*

Proof. The proof is slightly harder than the previous one; it is given in appendix. Note that unlike b and b', B and B' cannot be compared in general. □

10. Parametrised coinduction

Recall the coinductive proof principle, as provided by Knaster-Tarski's theorem:

$$\frac{x \leqslant y \leqslant by}{x \leqslant \nu b}$$

This approach has an important drawback: the need to define the coinductive predicate (y) up-front. This does not match the standard practice, where the coinductive predicate is obtained incrementally from x, by progressively extending it until it becomes a post-fixpoint. In the context of a paper proof, one can always gather the final coinductive predicate a posteriori, to display it at the beginning of the proof. In the context of interactive formal proofs, this becomes really inconvenient.

To solve this problem, Hur et al. proposed to use *parametrised coinduction* [11]. The trick consists in defining an auxiliary function $G_b : [X \to X]$ with the following properties:

$$G_b \bot = \nu b \tag{25}$$

$$G_b x = b(x \vee G_b x) \tag{26}$$

$$y \leqslant G_b(y \vee x) \;\Rightarrow\; y \leqslant G_b x \tag{27}$$

Concretely, they define $G_b x$ as the greatest fixpoint of the function $z \mapsto b(z \vee x)$. Intuitively, $G_b x$ represents what can be deduced using x as a coinduction hypothesis. When x is empty, we get the greatest fixpoint of b (25), this is how one initialises a proof by parametric coinduction. The second equation (26) makes it possible to unfold the bisimulation game: after having played b, one can either use the coinduction hypothesis (x), or continue the game $(G_b x)$. Implication (27) makes it possible to accumulate knowledge: to prove that y can be deduced from x, one can add y to the current set of coinductive assumptions. Note that we do no have $y \leqslant G_b y$: implication (27) would obviously be wrong otherwise. Instead, to get access to the coinductive assumptions, one needs to go through b (the bisimulation game) at least once, using the second equation (26). As explained by Hur et al., this corresponds to having a semantic guardedness check for corecursive definitions.

Hur et al. also show how to use up-to techniques with parametric coinduction, using the greatest respectful function (which coincides with t, according to Section 9). As in the present paper, the idea is to switch to $b^\dagger = bt$, and thus they use $G_{b^\dagger}$ rather than G_b. To be able to freely use up-to techniques in incremental proofs, they prove a fourth equation [11, Theorem 13]:

$$G_{b^\dagger} = tG_{b^\dagger} \tag{28}$$

Surprisingly, we actually have

Theorem 10.1. $G_{b^\dagger} = b^\dagger$.

(This result is proved in the appendix; it is a simple consequence of Theorem 10.2 below. Note that $G_b \neq b$ in general: the companion plays a crucial role in the above theorem. Also note that with such a characterisation, Equation (28), i.e., [11, Theorem 13], becomes a simple corollary: $tG_{b^\dagger} = tbt$ and $bt \leqslant tbt \leqslant btt = bt$.)

So we do not need the machinery of the $G_{b^\dagger}$ function to implement parametric coinducion; it is already provided by the companion. Following this idea, we can actually present parametric coinduction with a finer granularity.

Let us first prove the following counterpart to (27):

Theorem 10.2. *For all $x, y \in X$, if $y \leqslant bt(y \vee x)$ then $y \leqslant tx$.*

Proof. Assume $y \leqslant bt(y \vee x)$ (H), and let $f : z \mapsto \bigvee_{x \leqslant z} y$. This function maps the points above x to y and all other points to the bottom element. In particular, $fx = y$, so that we have to show $fx \leqslant tx$. We show a bit more, namely, that $f \leqslant t$.

$$\frac{y \leqslant t\bot}{y \leqslant \nu b}\ \textsc{Init} \qquad \frac{y \leqslant x}{y \leqslant tx}\ \textsc{Done}$$

$$\frac{y \leqslant ftx \qquad f \leqslant t}{y \leqslant tx}\ \textsc{Up to } f \qquad \frac{y \leqslant bt(y \vee x)}{y \leqslant tx}\ \textsc{CoInd}$$

Figure 4. A proof system for parametric enhanced coinduction.

To this end, we use second-order coinduction up-to, and we prove $f \leqslant B^\dagger f$, i.e, $fb \leqslant bTf$. Let $z \in X$. If $x \not\leqslant bz$, then $fbz = \bot \leqslant b(Tf)z$. Otherwise, assume $x \leqslant bz$ (H'); we have

$$\begin{aligned}
fbz &= y && \text{(by definition of } f \text{ and } (H'))\\
&\leqslant bt(y \vee x) && \text{(by } (H))\\
&= bt(fx \vee x) && \text{(by definition of } f)\\
&\leqslant bt(fbz \vee bz) && \text{(by } (H'))\\
&= b(t(fb \vee b))z
\end{aligned}$$

We easily check that $t(fb \vee b) \leqslant Tf$ using Proposition 6.4, so that we have $fb \leqslant bTf$, as required. □

Intuitively, tx contains everything that can safely be deduced from x, not necessarily in a guarded way. In particular, x can be deduced from tx. Instead, $btx = G_{b\dagger}x$ is more restrictive and corresponds to guarded deductions only: we do not have $x \leqslant btx$ in general. With this intuition, the above proposition reads as follows: to deduce y from x, one can assume y provided one switches to guarded deductions.

This leads us to the "proof system" given in Figure 4. The four rules are valid: if their premises hold, so do their conclusion. The first one is for initialisation: to prove that y is below the greatest fixpoint, deduce it from the empty context. The second rule is an axiom rule: if y belongs to the context x, then we can deduce y. The third one makes it possible to use any enhancement known to be below the companion. By stacking uses of this rule, one can perform inductive (or equational) reasoning. The fourth rule is just Theorem 10.2: it corresponds to an actual coinductive step. Also note that since $b \leqslant t$, the third rule can be used to play one step of the bisimulation game, without storing the current value of y in the context. Doing so corresponds to using Equation (26) from Hur et al.' formalism.

To give an example, let us revisit the example from Section 4. We wanted to prove that $A \sim B$, and we guessed that the relation $\mathcal{S} \triangleq \{\langle A, B\rangle, \langle C, D\rangle\}$ could be used as a bisimulation candidate, thanks to several enhancements. With the proof system from Figure 4, we can give a parametric-style proof, where we do not guess the bisimulation candidate in advance.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle A, B\rangle \in \mathcal{S}}{\langle A, B\rangle \in \mathbf{t}\mathcal{S}}\ (\textsc{Done}) \qquad \dfrac{\langle C, D\rangle \in \mathcal{S}}{\langle C, D\rangle \in \mathbf{t}\mathcal{S}}\ (\textsc{Done})}{\langle A|C, B|D\rangle \in \mathbf{t}\mathcal{S}}\ (\textsc{Up to } \mathbf{c}^| \leqslant \mathbf{t})}{\langle C, D\rangle \in \mathbf{bt}\mathcal{S}}\ (\textsc{Def. of } \mathbf{b})}{\langle C, D\rangle \in \mathbf{t}\langle A, B\rangle}\ (\textsc{CoInd})}{\langle D, C\rangle \in \mathbf{t}\langle A, B\rangle}\ (\textsc{Up to } \mathbf{i} \leqslant \mathbf{t})}{\langle b.D, b.C\rangle \in \mathbf{t}\langle A, B\rangle}\ (\textsc{Up to } \mathbf{c}^\cdot \leqslant \mathbf{t})}{\langle A, B\rangle \in \mathbf{bt}\langle A, B\rangle}\ (\textsc{Def. of } \mathbf{b})}{\langle A, B\rangle \in \mathbf{t}\bot}\ (\textsc{CoInd})}{A \sim B}\ (\textsc{Init})$$

As previously, the required enhancements do not need to be declared up-front, they are extracted from the companion using the third rule, when needed.

A similar proof could be obtained using Hur et al.' formalism, but we think that the proof system from Figure 4 is cleaner as it properly separates concerns: the main judgement is of the shape $x \leqslant ty$: "deduce x from y" either inductively or coinductively. The guardedness constraints appear only right after a coinductive step (rule CoInd), where the judgement temporarily takes the shape $x \leqslant bty$: "deduce x from y in a guarded way". In such a situation, after playing the game once, i.e., going through the definition of b, we get back to the original judgement.

Lastly, we left aside the proofs of $\mathbf{c}^{\cdot}, \mathbf{c}^{|}, \mathbf{i} \leqslant \mathbf{t}$ in the above example, because we obtained them once and for all in Sections 4 and 8.1. Note however that the third rule (Up to) allows us to jump to the next level in the middle of a proof: since $t = \nu B = T\bot$, one can fulfill its second premise by using the same proof system. In fact, by the results of Section 6, these four rules cover not only enhanced coinduction, but also the enhancements themselves. Nothing prevents us from continuing with the next level again, although we did not find any concrete application so far.

11. Directions for future work

GSOS is a rule format that was introduced to ensure congruence properties for bisimilarity [4]. We have recently shown that it also gives rise to a respectful contextual closure function: up-to context can always be used for GSOS specifications [5]. In light of the present results, we can deduce that such a closure is always contained in the companion in two ways: first by reusing the existing proofs of respectfulness and switching to the companion by Proposition 9.1; second, by an easy generalisation of our treatment of CCS (Section 8.1).

The later approach is rather intriguing from a categorical point of view. Indeed, GSOS specification can be seen abstractly as distributive laws [3, 33]. More precisely, when Σ is the functor corresponding to a term signature, and when $FX = (P_\omega X)^A$ is the functor whose coalgebras are the finitary branching LTS with labels in A, we have that a GSOS specification is exactly a distributive law

$$\Sigma(F \times Id) \Rightarrow FT$$

The fact that we have $F \times Id$ on the left makes it quite natural to consider respectfulness rather than compatibility when studying up-to-context techniques in this setting. The present results however suggest that there might be a more direct path, by using a categorical version of the companion.

References

[1] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. *Nord. J. Comput.*, 5(4):267–, 1998.

[2] S. Abramsky. The Lazy Lambda Calculus. In *Research Topics in Functional Programming*, pages 65–116. Addison Wesley, 1990.

[3] F. Bartels. Generalised coinduction. *Mathematical Structures in Computer Science*, 13(2):321–348, 2003.

[4] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. In *Proc. POPL*, pages 229–239. ACM, 1988.

[5] F. Bonchi, D. Petrisan, D. Pous, and J. Rot. Coinduction up-to in a fibrational setting. In *Proc. CSL-LICS*, page 20. ACM, 2014.

[6] M. Boreale and D. Sangiorgi. Bisimulation in name-passing calculi without matching. In *Proc. LICS*. IEEE Computer Society, 1998.

[7] K. Chaudhuri, M. Cimini, and D. Miller. A lightweight formalization of the metatheory of bisimulation-up-to. In *Proc. CPP*, pages 157–166. ACM, 2015.

[8] C. Fournet, J. Lévy, and A. Schmitt. An asynchronous, distributed implementation of mobile ambients. In *Proc. IFIP TCS*, volume 1872 of *LNCS*, pages 348–364. Springer, 2000.

[9] J. E. Hopcroft. An $n\log n$ algorithm for minimizing states in a finite automaton. Technical report, Stanford University, 1971.

[10] J. E. Hopcroft and R. M. Karp. A linear algorithm for testing equivalence of finite automata. Technical Report 114, Cornell University, December 1971.

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

[12] A. Jeffrey and J. Rathke. Towards a theory of bisimulation for local names. In *Proc. LICS*, pages 56–66, 1999.

[13] A. Jeffrey and J. Rathke. A theory of bisimulation for a fragment of concurrent ML with local names. *Theoretical Computer Science*, 323(1-3):1–48, 2004.

[14] B. Knaster. Un théorème sur les fonctions d'ensembles. *Annales de la Société Polonaise de Mathématiques*, 6:133–134, 1928.

[15] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In *Proc. POPL*, pages 141–152. ACM, 2006.

[16] S. B. Lassen. *Relational Reasoning about Functions and Nondeterminism*. PhD thesis, Department of CS, University of Aarhus, 1998.

[17] X. Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[18] J.-M. Madiot, D. Pous, and D. Sangiorgi. Bisimulations up-to: Beyond first-order transition systems. In *Proc. CONCUR*, volume 8704 of *LNCS*, pages 93–108. Springer, 2014.

[19] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[20] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I/II. *Information and Computation*, 100(1):1–77, 1992.

[21] J. Å. Pohjola and J. Parrow. Bisimulation up-to techniques for psi-calculi. In *Proc. CPP*, pages 142–153. ACM, 2016.

[22] D. Pous. Complete lattices and up-to techniques. In *Proc. APLAS*, volume 4807 of *LNCS*, pages 351–366. Springer, 2007.

[23] D. Pous. *Techniques modulo pour les bisimulations*. PhD thesis, École Normale Supérieure de Lyon, February 2008.

[24] D. Pous and D. Sangiorgi. *Advanced Topics in Bisimulation and Coinduction*, chapter about "Enhancements of the coinductive proof method". Cambridge University Press, 2011.

[25] J. Rot and M. M. Bonsangue. Combining bialgebraic semantics and equations. In *Proc. FOSSACS*, volume 8412 of *LNCS*, pages 381–395. Springer, 2014.

[26] J. J. M. M. Rutten. A coinductive calculus of streams. *Mathematical Structures in Computer Science*, 15(1):93–147, 2005.

[27] D. Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8:447–479, 1998.

[28] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[29] J. Sevcík, V. Vafeiadis, F. Z. Nardelli, S. Jagannathan, and P. Sewell. Compcerttso: A verified compiler for relaxed-memory concurrency. *Journal of the ACM*, 60(3):22, 2013.

[30] E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theoretical Computer Science*, 375(1-3):169–192, 2007.

[31] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *Journal of the ACM*, 54(5), 2007.

[32] A. Tarski. A Lattice-Theoretical Fixpoint Theorem and its Applications. *Pacific Journal of Mathematics*, 5(2):285–309, June 1955.

[33] D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS*, pages 280–291. IEEE Computer Society, 1997.

A. Omitted proofs

Proof for Remark 3.8. First note that the function mapping a function to its companion is not monotone, so that the inequality $t \leqslant t^\star$ ($\star$) does not follow directly from $b \leqslant b^\dagger$. Instead we show that t is compatible for $b^\dagger$:

$$tb^\dagger = tbt \leqslant btt = b^\dagger t \qquad \text{(by (3))}$$

Similarly for the converse inequality, we show that $t^\star$ is compatible for b:

$$\begin{aligned} t^\star b &\leqslant t^\star b^\dagger && \text{(by (5))} \\ &\leqslant b^\dagger t^\star && \text{(by (3) for } b^\dagger\text{)} \\ &= btt^\star && \text{(by definition)} \\ &\leqslant bt^\star t^\star = bt^\star && \text{(by } (\star) \text{ and idempotence of } t^\star\text{)} \end{aligned}$$

The second equality follows:

$$b^{\dagger\dagger} = b^\dagger t^\star = b^\dagger t = btt = bt = b^\dagger \qquad \square$$

Proof of Proposition 9.2. First recall from Proposition 9.1 and its proof that $t = t'$ and $b \leqslant b't$ (**). Let S be either T or T'; we first show

$$BS = B'S \tag{29}$$

Let $f, g : [X \to X]$, we have

$$\begin{aligned} f \leqslant B'Sg &\Leftrightarrow fb' \leqslant b'Sg && \text{(by (13))} \\ &\Leftrightarrow fb' \leqslant bSg && \text{(because } b \leqslant Sg\text{)} \\ &\Leftarrow fb \leqslant bSg && \text{(because } b' \leqslant b\text{)} \\ &\Leftrightarrow f \leqslant BSg && \text{(by (13))} \end{aligned}$$

(in the second step, we use $b \leqslant b't$ to deduce $b \leqslant S$ when $S = T'$.) Thus $BS \leqslant B'S$. Conversely, suppose that $f \leqslant B'Sg$, i.e., $fb' \leqslant bSg$. Then we have

$$\begin{aligned} fb &\leqslant fb't && \text{(by (**))} \\ &\leqslant b(Sg)t && \text{(by assumption)} \\ &\leqslant bSg && \text{(because } t \leqslant Sg\text{)} \end{aligned}$$

whence $f \leqslant BSg$, and consequently $B'S \leqslant BS$. Equation (29) is thus proved.

Now to prove $T = T'$, we prove that T is compatible for B:

$$TB' \leqslant TB'T = TBT = BT = B'T$$

and that T' is compatible for B:

$$T'B \leqslant T'BT' = T'B'T' = B'T' = BT'$$

We finally conclude: $B'T' = B'T = BT$. $\square$

Proof of Theorem 10.1. First note that we can strengthen the conclusion of Theorem 10.2 into $y \leqslant btx$. Indeed, from the hypothesis ($y \leqslant bt(y \vee x)$) and the stated conclusion ($y \leqslant tx$) we deduce $y \leqslant bt(tx \vee x)$; then we check that $t(tx \vee x) = tx$.

Now recall that $G_{b^\dagger}x$ is the greatest fixpoint of the function $z \mapsto bt(z \vee x)$. By monotonicity of bt, btx is a post-fixpoint of this function: $btx \leqslant bt(btx \vee x)$; thus $btx \leqslant G_{b^\dagger}x$. To prove $G_{b^\dagger}x \leqslant btx$, we use the aforementioned refinement of Theorem 10.2 and the fact that $G_{b^\dagger}x$ is a post-fixpoint, i.e., $G_{b^\dagger}x \leqslant bt(G_{b^\dagger}x \vee x)$. $\square$