Towards 'up to context' reasoning about higher-order processes

Adrien Durier, Daniel Hirschkoff, Davide Sangiorgi

Towards 'up to context' reasoning about higher-order processes

Adrien Durier[a,b], Daniel Hirschkoff[a], and Davide Sangiorgi[b]

[a]ENS Lyon
[b]Università di Bologna

August 15, 2018

Abstract

Proving behavioural equivalences in higher-order languages is a difficult task, because interactions involve complex values, namely terms of the language. In coinductive (i.e., bisimulation-like) techniques for these languages, a useful enhancement is the 'up-to context' reasoning, whereby common pieces of context in related terms are factorised out and erased. In higher-order *process* languages, however, such techniques are rare, as their soundness is usually delicate and difficult to establish.

In this paper we adapt the technique of unique solution of equations, that implicitly captures 'up-to context' reasoning, to the setting of the Higher-order π-calculus. Equations are written and solved with respect to normal bisimilarity, chosen both because of its efficiency — its clauses do not require universal quantifications on terms supplied by the external observer — and because of the challenges it poses on the 'up-to context' reasoning and that already show up when proving its congruence properties.

Keywords: bisimilarity, unique solution of equations, termination, process calculi, higher-order

Introduction

Bisimilarity is employed to define behavioural equivalences and reason about them. Finding its origins in concurrency theory, bisimilarity is now widely used also in other areas, as well as outside Computer Science.

We focus in this paper on *weak* bisimilarity, whereby we abstract from internal moves of terms (as opposed to the *strong* version, which makes no distinction between internal moves and transitions corresponding to interactions with the environment). Weak bisimilarity, and, more generally, weak equivalences, are more relevant in practice, because they allow one to compare programs which produce the same result with different numbers of reduction steps.

One of the main reasons for the success of bisimilarity is its coinductive proof method — the bisimulation proof method — whereby two terms are deemed bisimilar if there is a bisimulation relation containing them as a pair. An active topic of research studies proof methods for bisimilarity. The best known such strand is about enhancements of the bisimulation proof method [PS11]. In such techniques, called *up-to techniques*, the derivatives of two terms in the bisimulation game can be rewritten and manipulated, with the goal of working with candidate relations that need not be bisimulations — simply contained in a bisimulation. Among the most powerful such forms of enhancement is the *up-to context* technique, that allows one to exploit linguistic algebraic structures so to remove a common context in the derivative terms, and then requires the resulting terms to be related (the context may have multiple holes, in which case the tuples of resulting terms should be componentwise related). It is rarely the case that the derivatives terms explicitly exhibit a common context; usually they have to be 'massaged' (i.e., by applying some algebraic laws) in order to bring up such a common context. A common way of achieving this is by combining up-to context with *up-to expansion* (expansion is a preorder that refines bisimilarity); other combinations have also been used.

Up-to context techniques can be particularly effective in higher-order languages (where terms of the language are first-order values and may thus instantiate variables), such as the λ-calculi, Higher-order π-calculi, Ambients (e.g., [Las98a, Las98b, KW06, SW01, MN05]). Unfortunately, in such calculi proving the soundness of up-to context techniques can be surprisingly hard. Even in pure λ-calculi, and for the most basic form of bisimilarity (e.g., call-by-name or call-by-value λ-calculus and Ambramsky's applicative bisimilarity) long-standing open problems remain about soundness. Higher-order process calculi are the class of languages in which up-to context techniques in the literature are most scarce. Recently, techniques of this kind have been derived exploiting fully-abstract translations into first-order calculi (CCS-like or π-calculus-like) [MPS14]. However fully abstract translations are sensitive to the grammar of the language chosen: a modification to the grammar may break or prevent a fully abstract translation to be defined, or, at the very least, will require a careful re-examination of the full abstraction proof.

The topic of this paper is precisely proof techniques for bisimilarity in higher-order process calculi which allow one to reason up to contexts. We look for direct proofs of soundness, that do not rely on translations into first-order languages. To the best of our knowledge, the only such direct proofs are [MN05], for the Ambient calculus, and [SKS11] for the Higher-Order π-calculus (HOπ). The Ambient calculus represents a rather special case of higher-order calculus, for processes can move but cannot be communicated. Moving a process is quite different from communicating it as in HOπ: in the former case the process will always be run, immediately and exactly once; in the latter case, in contrast, the process may be copied, and it is the recipient of the process that decides when and where to run each copy. Thus the problems of soundness for up-to context only show up in a limited form in Ambients. The up-to context technique considered in [SKS11] is for environmental bisimilarity. This bisimilarity involves universal quantifications on processes supplied by the environment. Up-to context is essential for limiting the burden due to such quantifications. The contexts used have constraints and are disallowed in certain clauses (necessary for the soundness of the technique).

As a calculus we take HOπ and, as a form of bisimilarity, we consider *normal bisimilarity* [San96], for two main reasons. First, it is the most effective in proofs, as its clauses do not make use of additional universal quantifications with respect to ordinary bisimulation of CCS-like processes (other forms of bisimilarity, such as context bisimilarity or environmental bisimilarity, make use of universal quantifications in the input terms supplied by the external observer, as well as in other clauses). Second, precisely due to such lack of universal quantifications, the 'contextual' properties of normal bisimilarity are quite delicate. Even proving substitutivity with respect to basic operators such as parallel composition is hard (indeed, usually this is proved by relying on mappings onto other forms of bisimilarity or onto first-order calculi). No direct proofs of soundness of forms of up-to contexts exists (see also the discussion after Theorem 1.3 below).

In the paper we develop proof techniques for higher-order process calculi with the power of 'up-to contexts', by following the approach of *unique solutions of equations*. This bisimulation technique allows one to establish that two tuples of processes are componentwise bisimilar by establishing that they are solutions of the same system of equations. It was first proposed by Milner in the setting of CCS [Mil89], and has been used in verification tools based on algebraic reasoning [Ros10, BBR10, GM14]. Since uniqueness of solution does not hold for all equations (for instance the equation $X = \tau.X$ is satisfied, for weak equivalences, by any process), sufficient conditions must be stated to guarantee this property. In Milner's theorem [Mil89], the equations must be both *strongly guarded* and *sequential*: the variables of the equations may only be used underneath a visible prefix and preceded, in the syntax tree, only by the sum and prefix operators. This limits the expressiveness of the technique, since other operators, like parallel composition and restriction, are not allowed above the variables (and occurrences of these cannot, in general, be removed). Moreover, in several cases (e.g., languages for distributed systems or higher-order languages), the sum operator is absent, which means that the theorem cannot be adapted, or it would be essentially useless.

We therefore appeal to a development [DHS17] of Milner's theorem, in turn inspired by results by Roscoe in CSP [Ros97, Ros92] and that, in its basic form, essentially says that a guarded equation (or system of equations) whose infinite unfolding never produces a divergence has the unique-solution property. We transport the theory in [DHS17] to the setting of HOπ and normal bisimilarity. This also includes a refined version of the unique-solution theorem, in which we distinguish between different forms of divergence. Notably, we

can ignore divergences that appear after a finite number of unfoldings of the equations, called *innocuous* divergences.

In the paper, we recall the syntax and operational semantics of HOπ, the Higher-Order π-calculus, and show how we can adapt these definitions to write and solve systems of equations in HOπ. The structure of the proofs of the unique-solution theorems for HOπ in this paper is similar to that of the analogous theorems for CCS [DHS17]. We consider this as a positive outcome: the main objective of the current paper was to examine if and how the CCS proofs could be transported onto a higher-order setting. There are however noticeable differences in the proofs. For instance, often the proofs require reasoning on sequences of transitions. Now, in CCS the derivative of any transition is a process. In HOπ, in contrast, the derivative of an input is an abstraction, that needs to receive a process before becoming a process itself; similarly for output transitions, whose derivatives are concretions. This has also consequences on the reasoning about unfolding of equations. Similar issues with instantiation of terms occur in the definitions of bisimilarity in CCS and in HOπ, and are at the heart of the differences between them.

Paper outline. We recall the definition of HOπ and of normal bisimilarity in Section 1. Section 2 introduces systems of equations, and Sections 3 and 4 present our unique-solution techniques (Theorems 3.3 and 4.2), highlighting the difficulties that are specific to HOπ. We illustrate the use of our approach on an example in Section 5, and compare it with existing proof methods. Concluding remarks are given in Section 6.

1 The Higher-Order π-calculus

1.1 Processes and Transitions

In the syntax for HOπ we adopt, we use abstractions and concretions to represent input and output prefixes; we also use *first-order names*, i.e., names which carry nothing. These are opposed to *higher-order names*, i.e., names which are used to exchange processes. For the theory we shall develop, the presence of first-order names is not necessary, but makes the presentation of various results easier.

Thus, let $\mathcal{F}$ be the infinite set of first-order names, and $\mathcal{H}$ the infinite set of higher-order names. Then, $\overline{\mathcal{F}} \stackrel{\text{def}}{=} \{\overline{m} \mid m \in \mathcal{F}\}$, $\overline{\mathcal{H}} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in \mathcal{H}\}$, $\mathcal{N} \stackrel{\text{def}}{=} \mathcal{F} \cup \mathcal{H}$ and $\overline{\mathcal{N}} \stackrel{\text{def}}{=} \overline{\mathcal{F}} \cup \overline{\mathcal{H}}$. The special symbol τ, which does not occur in $\mathcal{N}$ or $\overline{\mathcal{N}}$, denotes a silent step. We let μ range over $\mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$, and ℓ range over $\mathcal{F} \cup \overline{\mathcal{F}} \cup \{\tau\}$ (the set of CCS-like actions). By convention, if $\ell \neq \tau$ then $\overline{\overline{\ell}} = \ell$. We use symbols x, y, z for names in $\mathcal{N}$; symbols m, n for names in $\mathcal{F}$; and symbols a, b, c for names in $\mathcal{H}$. We also assume an infinite set of *process variables*, ranged over by X, Y, Z, and a set *constant identifiers* (or simply *constants*) ranged over by K, H, to write recursively defined processes.

Definition 1.1. *The syntactic categories of our language and their grammar are:*

$$\begin{array}{rrcl} \textit{Processes} & P & ::= & a.F \mid \overline{a}.C \mid \ell.P \mid P_1 \mid P_2 \mid \boldsymbol{\nu} a\, P \mid X \mid F \circ P \mid \mathbf{0} \\ \textit{Abstractions} & F & ::= & (X)\,P \mid K \\ \textit{Concretions} & C & ::= & \boldsymbol{\nu} x\, C \mid \langle P_1 \rangle P_2 \\ \textit{Agents} & A & ::= & P \mid F \mid C \end{array}$$

A prefix $a.(X)\,P$ represents an input, in which the process received at a will replace X in P. Dually, $\overline{a}.\langle P_1 \rangle P_2$ is an output, in which P_1 is emitted at a and P_2 is the continuation. In an application $F \circ P$ process P is supposed to replace the formal parameter of the abstraction F. The remaining operators are the usual one of CCS and derived calculi. Symbols $P, Q, R, \ldots$ range over processes, F, G over abstractions, C, D over concretions, A, B over agents. Each constant K has a corresponding definition $K \stackrel{\triangle}{=} (X)\,P$ (as P may contain K, the behaviour may be recursive). Although in HOπ recursion can be derived (examples of this will be shown in Section 5), we take constants as primitives because they are useful when reasoning

about equations. The calculus is monadic — (higher-order) names carry exactly one name; correspondingly, abstractions are parametrised over exactly one value. We stick to monadicity for simplicity of presentation.

We often omit trailing occurrences of $\mathbf{0}$, therefore abbreviating $\overline{m}.\,\mathbf{0}$ as $\overline{m}$, and $\overline{a}\,.\,\langle Q\rangle\mathbf{0}$ as $\overline{a}\,.\,\langle Q\rangle$; we also omit the parameter in constants when it is not needed, therefore simply writing $K \stackrel{\triangle}{=} P$ and simply using K as a process. We shall also sometimes use a special form of constants, namely the replication $!P$, that intuitively stands for an infinite number of copies of P in parallel; it can be written as the constant $K_P \stackrel{\triangle}{=} P \mid K_P$.

An application redex $((X)\,P) \circ Q$ can be normalised as $P\{{}^{Q}/_{X}\}$. An agent is *normalised* if all such application redexes have been contracted. Although the full application $F \circ P$ is often convenient, when it comes to reasoning on behaviours it is useful to assume that all expressions are normalised, in the above sense. Thus in the remainder of the paper *we identify an agent with its normalised expression.* The application construct $F \circ P$ will play an important role in the treatment of equations in the following sections.

An abstraction $(X)\,P$ binds all free occurrences of X in P; similarly a restriction $\boldsymbol{\nu} x\,P$ binds the free occurrences of x in P. These binders give rise in the expected way to the definitions of alpha conversion, free (process) variables and free names of an agent A. An agent is *closed* if it has no free variable (it can have free names). We use a tilde to denote a tuple; all notations are extended to tuples componentwise. $P\{{}^{\widetilde{Q}}/_{\widetilde{X}}\}$ denotes the componentwise and simultaneous substitution of variables $\widetilde{X}$ with processes $\widetilde{Q}$ (where it is assumed that the members of $\widetilde{X}$ are distinct). We often abbreviate $\boldsymbol{\nu} x_1\ \ldots \boldsymbol{\nu} x_n\,A$ as $(\boldsymbol{\nu} x_1, \ldots, x_n)A$. In a statement, we shall say that a name is *fresh* to mean that it is different from any other name occurring in agents of the statement.

We shall only admit *standard concretions*, i.e., expressions $\boldsymbol{\nu}\widetilde{x}\,\langle P_1\rangle P_2$ where names in $\widetilde{x}$ are pairwise distinct and $\widetilde{x} \subseteq \mathsf{fn}(P_1)$. Indeed, the remaining concretions have little significance: in $\boldsymbol{\nu}\widetilde{x}\,\langle Q\rangle P$, by alpha conversion, names $\widetilde{x}$ can be assumed to be distinct; and if $x \notin \mathsf{fn}(Q) \cup \{\widetilde{x}\}$ then in $(\boldsymbol{\nu} x, \widetilde{x})\langle Q\rangle P$ the restriction on name x can be pushed inwards, resulting in the standard concretion $\boldsymbol{\nu}\widetilde{x}\,\langle Q\rangle(\boldsymbol{\nu} x\,P)$. In the following, we therefore assume that if $x \notin \mathsf{fn}(Q) \cup \{\widetilde{x}\}$, then $(\boldsymbol{\nu} x, \widetilde{x}\,)\langle Q\rangle P$ denotes $\boldsymbol{\nu}\widetilde{x}\,\langle Q\rangle(\boldsymbol{\nu} x\,P)$.

We wish to extend restriction to operate on abstractions, and (a form of) parallel composition to operate on abstractions and concretions:

Let $F = (X)\,Q$:

— if $X \notin \mathsf{fv}(P)$ then $F \mid P$ denotes $(X)\,(Q \mid P)$

(and similarly for $P \mid F$),

— $\boldsymbol{\nu} x\,F$ denotes $(X)\,\boldsymbol{\nu} x\,Q$;

if $C = \boldsymbol{\nu}\widetilde{x}\,\langle Q\rangle R$ then

if $\widetilde{x} \cap \mathsf{fn}(P) = \emptyset$ then $C \mid P$ denotes $\boldsymbol{\nu}\widetilde{x}\,\langle Q\rangle(R \mid P)$

(and similarly for $P \mid C$).

We now present the operational semantics of the calculus. First, we define an operation of pseudo-application between an abstraction $F = (X)\,P$ and a concretion $C = \boldsymbol{\nu}\widetilde{x}\,\langle Q\rangle R$. By alpha conversion, we can assume that $\widetilde{x} \cap \mathsf{fn}(F) = \emptyset$, and then we set

$$C \bullet F \stackrel{\text{def}}{=} \boldsymbol{\nu}\widetilde{x}\,(R \mid P\{{}^{Q}/_{X}\})$$

and, symmetrically,

$$F \bullet C \stackrel{\text{def}}{=} \boldsymbol{\nu}\widetilde{x}\,(P\{{}^{Q}/_{X}\} \mid R)\,.$$

The operational semantics of the calculus is reported in Table 1. We have omitted the symmetric forms

Prefix	$\mu.\, A \xrightarrow{\mu} A$		
Parallelism	$P_1 \xrightarrow{\mu} A$	implies	$P_1 \mid P_2 \xrightarrow{\mu} A \mid P_2$
First-order communication	$P_1 \xrightarrow{m} P_1'$		
	$P_2 \xrightarrow{\overline{m}} P_2'$	implies	$P_1 \mid P_2 \xrightarrow{\tau} P_1' \mid P_2'$
Higher-order communication	$P_1 \xrightarrow{a} F$		
	$P_2 \xrightarrow{\overline{a}} C$	implies	$P_1 \mid P_2 \xrightarrow{\tau} F \bullet C$
Restriction	$P \xrightarrow{\mu} A,\ \mu \notin \{x, \overline{x}\}$	implies	$\boldsymbol{\nu} x\, P \xrightarrow{\mu} \boldsymbol{\nu} x\, A$
Constants	$F \circ P \xrightarrow{\mu} A$		
	$K \stackrel{\triangle}{=} F$	implies	$K \circ P \xrightarrow{\mu} A$

Table 1: The transition system

of the parallelism and communication rules. The following forms of judgements are introduced:

$$\begin{array}{ll} P \xrightarrow{a} F & \text{(higher-order input transition at port } a\text{)} \\ P \xrightarrow{\overline{a}} C & \text{(higher-order output transition at port } a\text{)} \\ P \xrightarrow{\ell} Q & \text{(first-order transition)} \end{array}$$

where P, Q, F, C are agents. In turn, a first-order transition can be a first-order input (if $\ell \in \mathcal{F}$), a first-order output (if $\ell \in \overline{\mathcal{F}}$), or an interaction (if $\ell = \tau$).

In the remainder of the paper we work up to alpha conversion; thus "$=$" denotes syntactic equality up to alpha conversion. We shall normally put enough brackets in the expressions so to avoid precedence ambiguities among the operators. However, to reduce the number of brackets, in a few places we shall assume the following syntactic rules: substitutions and notations "$\bullet$" and "$\circ$" have the highest syntactic precedence; the abstraction and concretion constructs the lowest; parallel composition has weaker precedence than the other process constructs. For instance, $\langle P \rangle !m.\, R \mid Q$ stands for $\langle P \rangle ((!m.\, R) \mid Q)$, and $F \bullet C \mid Q\{^R/_X\}$ stands for $(F \bullet C) \mid (Q\{^R/_X\})$.

If $\mathcal{R}$ is a relation on processes, we write $P\ \mathcal{R}\ Q$ to denote that $(P, Q) \in \mathcal{R}$. Moreover, $\mathcal{R}_1\mathcal{R}_2$ is the composition of the two relations $\mathcal{R}_1$ and $\mathcal{R}_2$.

1.2 Normal bisimulation

We only discuss *weak* equivalences, i.e., equivalences that abstract from internal steps and that, practically, are the most relevant ones. To introduce them, we first define weak transitions. Thus $\Longrightarrow$ is the reflexive and transitive closure of $\xrightarrow{\tau}$, and $P \stackrel{\mu}{\Longrightarrow} A$ holds if there is P' such that $P \Longrightarrow P'$ and $P' \xrightarrow{\mu} A$ (note that weak transitions are defined using the 'delay semantics', as we do not allow τ-steps after the μ-transition). Finally, $P \stackrel{\widehat{\mu}}{\Longrightarrow} A$ is $P \Longrightarrow A$ if $\mu = \tau$ and $P \stackrel{\mu}{\Longrightarrow} A$ otherwise.

An arguably natural notion of bisimulation for higher-order processes is *context bisimulation*; it exploits the duality between abstractions and concretions, so to test an abstraction with all possible concretions it can be used with, and conversely. Its bisimulation clauses on higher-order inputs and outputs are therefore as follows, if $\mathcal{R}$ is a context-bisimulation candidate:

1. whenever $P \xrightarrow{a} F$, there exists G s.t. $Q \stackrel{a}{\Longrightarrow} G$ and $C \bullet F \mathcal{R} C \bullet G$, for all closed concretions C;

2. whenever $P \xrightarrow{\overline{a}} C$, there exists D s.t. $Q \stackrel{\overline{a}}{\Longrightarrow} D$ and $F \bullet C \mathcal{R} F \bullet D$, for all closed abstractions F.

Surprisingly, the universal quantifications on concretions and abstractions supplied by the external observer may be removed using a special kind of concretion and of abstraction, namely the concretion $\langle \overline{m}.\, \mathbf{0} \rangle \mathbf{0}$ and the abstraction $(X)\, !m.\, X$, where m is supposed to be a name fresh in the tested processes. In the former concretion, the continuation is null and therefore we can simply use the process $\overline{m}.\, \mathbf{0}$ (or even just $\overline{m}$), called a *trigger* and abbreviated $\mathtt{Tr}_m$. Similarly we write $\mathtt{Ab}_m$ as an abbreviation for $(X)\, !m.\, X$. We thus obtain *normal bisimulation*.

Definition 1.2 (normal bisimulation)**.** *A relation $\mathcal{R} \subseteq \mathrm{Pr} \times \mathrm{Pr}$ is a* normal simulation *if $P \mathcal{R} Q$ implies, for $m \notin fn(P, Q)$:*

1. *whenever $P \xrightarrow{\ell} P'$, there exists Q' s.t. $Q \stackrel{\widehat{\ell}}{\Longrightarrow} Q'$ and $P' \mathcal{R} Q'$;*

2. *whenever $P \xrightarrow{a} F$, there exists G s.t. $Q \stackrel{a}{\Longrightarrow} G$ and $F \circ \mathtt{Tr}_m \mathcal{R} G \circ \mathtt{Tr}_m$;*

3. *whenever $P \xrightarrow{\overline{a}} C$, there exists D s.t. $Q \stackrel{\overline{a}}{\Longrightarrow} D$ and $C \bullet \mathtt{Ab}_m \mathcal{R} D \bullet \mathtt{Ab}_m$.*

A relation $\mathcal{R}$ is a normal bisimulation*, in symbols $\approx$-bisimulation, if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are normal simulations. We say that P and Q are* normal bisimilar*, in symbols $P \approx Q$, if $P \mathcal{R} Q$, for some $\approx$-bisimulation $\mathcal{R}$.*

In clauses (2) and (3) of Definition 1.2 it is enough to pick *some* fresh name m, since the specific choice of the fresh name does not affect the equivalence of the resulting processes. The extension of $\approx$ to closed abstractions and open agents is defined similarly, only employing fresh triggers. For closed concretions C and D we set $C \approx D$ if $C \bullet \mathtt{Ab}_m \approx D \bullet \mathtt{Ab}_m$, for some fresh m.

Theorem 1.3. *Normal bisimilarity is a congruence relation in HOπ.*

The proof in [San96] derives the congruence of normal bisimilarity from congruence of context bisimilarity. In doing so, an auxiliary relation, triggered bisimilarity, is used for an intermediate characterisation. We are not aware of a simpler and direct proof of congruence for normal bisimulation, using the bisimulation proof method. The hard case is that of parallel composition. Usually, substitutivity of a bisimilarity with respect to an operator is established using, as a candidate relation, the closure of the bisimilarity under the operator itself (possibly enhanced with an auxiliary operator such as restriction), and using some 'bisimulation up-to' techniques to simplify the reasoning.

For normal bisimilarity and parallel composition, we have to prove that $P \approx Q$ implies $P \mid T \approx Q \mid T$ for all T. In the case of a higher-order interaction between P and T, the hypotheses at hand can be used to show the existence of a matching interaction between Q and T, but this is not sufficient to conclude. Normal bisimilarity guarantees that P and Q remain related after a higher-order input only when a trigger process is received; however T may transmit an arbitrary process. Replacing a trigger with an arbitrary process is possible as an algebraic law. However the replacement is unsound within up-to techniques. The natural up-to technique would be the 'up-to expansion' technique (*expansion* is a behavioural preorder, finer than weak bisimilarity, capturing an idea of 'process efficiency'). For the technique to be sound, however, the expansion preorder must be applied in a specific direction, whereas the algebraic transformation needed for the replacement of a trigger would require the opposite direction. Similar problems exist on interactions between P and T, and between Q and T when P and Q perform higher-order outputs.

2 Systems of Equations in HOπ

In CCS, uniqueness of solutions of equations [Mil89] intuitively says that if a context C obeys certain conditions, then all processes P that satisfy the equation $P \approx C[P]$ are bisimilar with each other.

Since in HOπ, in contrast with CCS, processes may be communicated, it is useful to have parametrised equations — much like the difference between ordinary constants of CCS and the parametrised constants of HOπ. Therefore the solution of an HOπ equation is an abstraction (as opposed to a process, as for a CCS equation). We use bold letters $\mathbf{X}, \mathbf{Y}, \ldots$ for the variables needed for writing equations, and called *equation variables*. The body of an equation is an abstraction possibly containing equation variables. As elsewhere in the paper, we stick to monadic abstractions.

We call *extended HOπ* the HOπ syntax of Section 1 extended with equation variables. To make the distinction between the ordinary syntax of HOπ and the syntax of extended HOπ, we use bold letters to range over the syntactic categories of the latter. Thus, in extended HOπ, $\mathbf{P}, \mathbf{Q}$ stand for *extended processes*, $\mathbf{F}$ for *extended abstractions*, $\mathbf{C}$ for *extended concretions*, and $\mathbf{A}$ for extended agents (the word 'extended' is used here to denote the fact that equation variables may occur). The grammar for $\mathbf{F}$ is:

$$\mathbf{F} \quad ::= \quad (X)\,\mathbf{P} \;\Big|\; K \;\Big|\; \mathbf{X}.$$

Extended agents can be either extended processes, extended concretions or extended abstractions.

Definition 2.1. *Assume that, for each i of a countable indexing set I, we have variables $\mathbf{X}_i$, and abstractions $\mathbf{F}_i$ possibly containing such variables. Then $\{\mathbf{X}_i = \mathbf{F}_i\}_{i\in I}$ is a* system of HOπ equations. *(There is one equation for each variable $\mathbf{X}_i$.)*

Example 2.2. *We report some examples of systems of equations, whose solutions are then discussed in Example 2.5:*

1. $\mathbf{X} = (Y)\,(\mathbf{X} \circ Y)$.
2. $\mathbf{X} = (Y)\,(\mathbf{X} \circ Y \mid Y)$.
3. $\mathbf{X} = (X)\,a.\,(Y)(X \mid Y)$ *(where the variable $\mathbf{X}$ of the equation does not occur in the body)* .
4. $\begin{array}{ll} \mathbf{X} & = (X)\,d.\,(Z)\,\mathbf{Y} \circ Z \\ \mathbf{Y} & = (Z)\,r.\,(Z \mid \mathbf{Y} \circ Z) \end{array}$

If the parameter of the equation is not important, we omit it, as in the equation $\mathbf{X} = a.\,(X)\overline{b}.\langle X\rangle\mathbf{X}$, whose solution is a link process that repeatedly receives a process at a and emits it at b.

We write $\mathbf{A}[\widetilde{F}]$ for the expression resulting from $\mathbf{A}$ by replacing each variable $\mathbf{X}_i$ with the abstraction F_i, assuming $\widetilde{F}$ and $\widetilde{\mathbf{X}}$ have the same length.

Remark 2.3. To avoid issues with restriction, we take the replacement in $\mathbf{A}[\widetilde{F}]$ to be a substitution, not a syntactic replacement (i.e., in the substitution, free names of an F_i may not be bound by a restriction in $\mathbf{A}$). An alternative would have been to impose abstractions without free names; this would however introduce issues related to name mobility and make the proofs of the main results more complex.

As for the plain syntax, so in the extended syntax we assume a normalisation of application redexes; thus any subterm of the form $((X)\,P) \circ Q$ should be thought of as an abbreviation for $P\{^{Q}/_{X}\}$.

Definition 2.4. *Suppose $\{\mathbf{X}_i = \mathbf{F}_i\}_{i\in I}$ is a system of equations:*

- *$\widetilde{F}$ is a* solution of the system of equations for $\approx$ *if for each i it holds that $F_i \approx \mathbf{F}_i[\widetilde{F}]$;*
- *it has* a unique solution for $\approx$ *if whenever $\widetilde{F}$ and $\widetilde{G}$ are both solutions for $\approx$, then $\widetilde{F} \approx \widetilde{G}$.*

Example 2.5. *Consider the equations from Example 2.2:*

1. *the equation $\mathbf{X} = (Y)\,(\mathbf{X} \circ Y)$ is satisfied by any closed abstraction of the form $(Y)\,P$.*
2. *In $\mathbf{X} = (Y)\,(\mathbf{X} \circ Y \mid Y)$, the abstraction $(Y)\,(P \mid !Y)$ is solution, for any P.*
3. *The body of the equation $\mathbf{X} = (X)\,a.\,(Y)(X \mid Y)$ does not use the variable $\mathbf{X}$, hence the only solution is precisely its defining abstraction $(X)\,a.\,(Y)(X \mid Y)$.*

4. *The system of equations*

$$\mathbf{X} = (X)\, d.\, (Z)\, \mathbf{Y} \circ Z$$
$$\mathbf{Y} = (Z)\, r.\, (Z \mid \mathbf{Y} \circ Z)$$

has a unique solution. The abstraction for $\mathbf{X}$ *receives a process at* d*, and then makes it available at* r*, so that any output at* r *will start a copy of the received process.*

To show that a system of equations has a unique solution, we manipulate the *unfoldings* of said system of equations. The unfolding of $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$, written $\widetilde{\mathbf{F}}[\widetilde{\mathbf{F}}]$, is the system obtained by replacing each $\mathbf{X}_i$ with $\mathbf{F}_i$ in $\widetilde{\mathbf{F}}$. As when instantiating using (non extended) abstractions, this operation is a substitution, not syntactic replacement, and we assume normalisation of application redexes.

The operation of unfolding can be iterated, giving rise to the notion of finite unfolding:

Definition 2.6 (Finite unfoldings)**.** *Given a system of equations* $\{\mathbf{X}_i = \mathbf{F}_i\}_{i \in I}$*, its* finite unfoldings $\mathbf{F}_i^n$ *are defined recursively, for any* $i \in I$*:*

- $\mathbf{F}_i^0$ *is defined as* $\mathbf{X}_i$
- $\mathbf{F}_i^{n+1}$ *is defined as* $\mathbf{F}_i^n[\widetilde{\mathbf{F}}]$

Accordingly, $\mathbf{F}_i^1$ is defined as $\mathbf{F}_i$. We write $\widetilde{\mathbf{F}}^n$ for the tuple $\{\mathbf{F}_i^n\}_{i \in I}$.

Definition 2.7 (Syntactic solution)**.** *Given a system of equations* $\{\mathbf{X}_i = \mathbf{F}_i\}_{i \in I}$*, its* syntactic solution *is defined as the tuple of mutually recursive constants* $\widetilde{K_{\mathbf{F}}}$ *(there is one* $K_{\mathbf{F},i}$ *for each* $i \in I$ *), where the defining equations are* $K_{\mathbf{F},i} \triangleq \mathbf{F}_i[\widetilde{K_{\mathbf{F}}}]$*.*

The syntactic solution of a system of equations can be seen as the result of the infinite unfolding of the equations defining the system.

Example 2.8. *We show the syntactic solutions of some of the equations in Example 2.5.*

1. *The syntactic solution of the equation* $\mathbf{X} = (Y)\, \mathbf{X} \circ Y$ *is the constant* $K \triangleq (Y)\, (K \circ Y)$*. This recursive process actually behaves as* $\mathbf{0}$*.*
2. *The syntactic solution of the equation* $\mathbf{X} = (Y)\, (\mathbf{X} \circ Y \mid Y)$ *is the constant* $K \triangleq (Y)\, (K \circ Y \mid Y)$*, which receives a process and runs infinitely many copies of it.*
3. *The syntactic solution of the equation* $\mathbf{X} = (X)\, a.\, (Y)(X \mid Y)$ *is the constant* $K \triangleq (X)\, a.\, (Y)(X \mid Y)$ *— as the equation does not use equation variables, the expression of the equation is the same as the expression defining the constant.*

The lemma below shows a form of commutativity on the order in which replacements are made, when unfolding equations and instantiating its variables.

Lemma 2.9. *For any extended agent* $\mathbf{A}$*, and extended abstractions* $\widetilde{\mathbf{F}}$ *and* $\widetilde{\mathbf{F}}'$*, we have*

$$\mathbf{A}[\widetilde{\mathbf{F}}][\widetilde{\mathbf{F}}'] = \mathbf{A}[\widetilde{\mathbf{F}}[\mathbf{F}']] \ .$$

Note that in the above lemma, $\widetilde{\mathbf{F}}$ and $\widetilde{\mathbf{F}}'$ need not have the same length. A special case of the above lemma is when $\widetilde{\mathbf{F}}'$ are simple (i.e., non extended) abstractions, say $\widetilde{F'}$. In this case we obtain $\mathbf{A}[\widetilde{\mathbf{F}}][\widetilde{F'}] = \mathbf{A}[\widetilde{\mathbf{F}}[\widetilde{F'}]$. This observation also holds for other results below, which are often specialised with simple abstractions.

Lemma 2.10. *For any extended agent* $\mathbf{A}$*, we have that* $\mathbf{A}[\widetilde{\mathbf{F}}]$ *is guarded if either* $\mathbf{A}$ *is guarded, or all extended abstractions* $\widetilde{\mathbf{F}}$ *are guarded.*

We use the SOS rules of Table 1 also on the syntax of extended HOπ. The following lemma relates transitions of terms of extended HOπ, possibly containing free equation variables, with terms resulting from instantiations of such variables.

Lemma 2.11.

1. *Given* $\mathbf{P}$ *an extended process and* $\mathbf{A}$ *an extended agent, if* $\mathbf{P} \xrightarrow{\mu} \mathbf{A}$*, then* $\mathbf{P}[\widetilde{\mathbf{F}}] \xrightarrow{\mu} \mathbf{A}[\widetilde{\mathbf{F}}]$*, for all extended abstractions* $\widetilde{\mathbf{F}}$*.*

2. *If* $\mathbf{P}$ *is a guarded extended process and* $\mathbf{P}[\widetilde{\mathbf{F}}] \xrightarrow{\mu} \mathbf{A}_0$*, then there is an extended agent* $\mathbf{A}$ *such that* $\mathbf{P} \xrightarrow{\mu} \mathbf{A}$ *and* $\mathbf{A}_0 = \mathbf{A}[\widetilde{\mathbf{F}}]$*.*

Both properties of the previous lemma are valid, as a special case, for transitions emanating from $\mathbf{P}[\widetilde{F}]$. When case 1 above holds, we call a transition $\mathbf{P}[\widetilde{\mathbf{F}}] \xrightarrow{\mu} \mathbf{A}[\widetilde{\mathbf{F}}]$ *an instance* of the transition $\mathbf{P} \xrightarrow{\mu} \mathbf{A}$.

The following lemma states basic properties about $\circ$ and $\bullet$; it is used implicitly in several places below.

Lemma 2.12.

- *For all* $\mathbf{F}$*,* $\widetilde{F}$*, and* Q*, we have* $(\mathbf{F} \circ Q)[\widetilde{F}] = \mathbf{F}[\widetilde{F}] \circ Q$*.*
- *For all* $\mathbf{C}$*,* $\widetilde{F}$ *and* F*, we have* $(\mathbf{C} \bullet F)[\widetilde{F}] = \mathbf{C}[\widetilde{F}] \bullet F$*.*

Recall that for a name m, a *trigger* at m is a process $\mathtt{Tr}_m \stackrel{\text{def}}{=} \overline{m}.\,\mathbf{0}$, often simply written $\overline{m}$, and that $\mathtt{Ab}_m \stackrel{\text{def}}{=} (X)!m.\,X$. Thus, given an extended concretion $\mathbf{C} = \langle \mathbf{P}_1 \rangle \mathbf{P}_2$ (resp. an extended abstraction $\mathbf{F} = (X)\ \mathbf{P}$), we have $\mathbf{C} \bullet \mathtt{Ab}_m = !m.\,\mathbf{P}_1 \mid \mathbf{P}_2$ (resp. $\mathbf{F} \circ \mathtt{Tr}_m = \mathbf{P}\{\overline{m}/X\}$).

Divergences in HOπ. In order to state the unique solution theorems in Sections 3 and 4, we need to introduce the notion of divergence in HOπ. A divergence in a process consists of a finite sequence of transitions (of any kind) followed by an infinite sequence of τ transitions. A diverging process is a process that contains a divergence. In other words, before embarking in a sequence of internal moves, a diverging process may perform some higher-order interactions with its environment. In order to account for these, we follow the definition of normal bisimulation (Definition 1.2), and instantiate agents with the special forms for triggers and abstractions (processes $\mathtt{Tr}_m$ and $\mathtt{Ab}_m$). By using such processes with freshly generated names, we avoid possible interferences with other transitions of the process, and do not break divergences.

We first introduce the notion of reduct, which then allows us to define divergences.

Definition 2.13 (Reducts)**.**

1. *We say that* $\mathbf{P}$ reduces to $\mathbf{P}'$*, written* $\mathbf{P} \rightarrowtriangle \mathbf{P}'$*, if one of the following holds:*

   (a) $\mathbf{P} \xrightarrow{\ell} \mathbf{P}'$*;*

   (b) $\mathbf{P}' = \mathbf{C} \bullet \mathtt{Ab}_m$ *for some fresh* m *and some* $\mathbf{C}$ *such that* $\mathbf{P} \xrightarrow{\overline{a}} \mathbf{C}$*;*

   (c) $\mathbf{P}' = \mathbf{F} \circ \mathtt{Tr}_m$ *for some fresh* m *and some* $\mathbf{F}$ *such that* $\mathbf{P} \xrightarrow{a} \mathbf{F}$*.*

   Relation $P \rightarrowtriangle P'$ *("*P *reduces to* P' *") is defined in the same way for processes that do not contain equation variables.*

2. *The* set of reducts *of an extended process* $\mathbf{P}$*, written* $\mathtt{red}(\mathbf{P})$*, is given by:*

$$\mathtt{red}(\mathbf{P}) \quad \triangleq \quad \bigcup_n \{\mathbf{P}_n \mid \mathbf{P} \rightarrowtriangle \mathbf{P}_1 \ldots \rightarrowtriangle \mathbf{P}_n \text{ for some } n \text{ and } \mathbf{P}_i\ (1 \leq i \leq n)\ \}.$$

   Again, the set of reducts *of a process* P*,* $\mathtt{red}(P)$*, is defined in the same way.*

3. *The set of* reducts of the unfoldings *of a system of equations* $\{\mathbf{X}_i = \mathbf{F}_i\}_{i\in I}$*, written* $\mathtt{red}_\omega(\widetilde{\mathbf{F}})$*, is defined as*

$$\mathtt{red}_\omega(\widetilde{\mathbf{F}}) \triangleq \bigcup_{n\in\mathbb{N}, i\in I, m \text{ fresh}} \mathtt{red}(\mathbf{F}_i^n \circ \mathtt{Tr}_m).$$

Definition 2.14 (Divergence)**.** *A process* P diverges *if there is* $P' \in \mathtt{red}(P)$ *such that* P' *can perform an infinite sequence of internal moves; i.e., there are processes* P_i*,* $i \geq 0$ *such that* $P' = P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} P_2 \xrightarrow{\tau} \ldots$.

3 Unique Solution Theorem

The proofs presented in this section are, in spirit, similar to the proofs in the case of CCS [DHS17]. Technically, they rely in an essential way on reasoning about the transitions of unfoldings of the equations, based on the observation that if F is a solution of the equation $\mathbf{X} = \mathbf{F}$, then F is also a solution of equation $\mathbf{X} = \mathbf{F}[\mathbf{F}]$, of $\mathbf{X} = \mathbf{F}[\mathbf{F}[\mathbf{F}]]$, and so on.

In analysing such transitions, there are several aspects which differ between CCS and HOπ. First, we rely on the notions of reducts and divergences (Definitions 2.13 and 2.14), which, as explained above, have to be tailored to HOπ. Moreover, input and output transitions yield abstractions and concretions, which need to be instantiated. This shows up in several places below, like in the following definition, in clauses 3 and 4, when we need to refer to processes in order to impose a condition involving $\approx$.

Definition 3.1. *A system of equations* $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$ protects its solutions *if, for all solution* $\widetilde{F}$*, and for all* $\mathbf{P}' \in \mathtt{red}_\omega(\widetilde{\mathbf{F}})$*, the following hold:*

1. *if* $\mathbf{P}'[\widetilde{F}] \Longrightarrow Q$ *for some* Q*, then there exists* $\mathbf{P}''$ *and* n *such that* $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \Longrightarrow \mathbf{P}''$ *and* $\mathbf{P}''[\widetilde{F}] \approx Q$.
2. *if* $\mathbf{P}'[\widetilde{F}] \stackrel{\ell}{\Longrightarrow} Q$ *for some* Q*, then there exists* $\mathbf{P}''$ *and* n *such that* $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{\ell}{\Longrightarrow} \mathbf{P}''$ *and* $\mathbf{P}''[\widetilde{F}] \approx Q$.
3. *if* $\mathbf{P}'[\widetilde{F}] \stackrel{a}{\Longrightarrow} F_0$*, for some* a *and some* F*, then there exists* $\mathbf{F}$ *and* n *such that* $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{a}{\Longrightarrow} \mathbf{F}_0$ *and* $(\mathbf{F}_0 \circ \mathtt{Tr}_m)[\widetilde{F}] \approx F_0 \circ \mathtt{Tr}_m$ *(for any* m *fresh).*
4. *if* $\mathbf{P}'[\widetilde{F}] \stackrel{\overline{a}}{\Longrightarrow} C$*, for some* a *and some* C*, then there exists* $\mathbf{C}$ *and* n *such that* $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{\overline{a}}{\Longrightarrow} \mathbf{C}$ *and* $(\mathbf{C} \bullet \mathtt{Ab}_m)[\widetilde{F}] \approx \mathbf{C} \bullet \mathtt{Ab}_m$ *(for any* m *fresh).*

Intuitively, a system of equations $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$ protect its solutions when transitions emanating from $\mathbf{P}[\widetilde{F}]$, where $\mathbf{P} \in \mathtt{red}_\omega(\widetilde{\mathbf{F}})$ and $\widetilde{F}$ is a solution, can be mimicked by transitions of $\mathbf{P}[\widetilde{\mathbf{F}}^n]$ for some n — i.e., by replacing the solutions with unfoldings of the equations.

Proposition 3.2. *A system of equations that protects its solutions has a unique solution for* $\approx$.

Proof. We have to show that given two solutions $\widetilde{F}$, $\widetilde{F'}$ of the system of equations $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$, and a fresh name m_0, it holds that $\widetilde{F} \circ \mathtt{Tr}_{m_0} \approx \widetilde{F'} \circ \mathtt{Tr}_{m_0}$.

With these hypotheses, we set $\widetilde{\mathbf{P}} = \widetilde{\mathbf{F}} \circ \mathtt{Tr}_{m_0}$, $\widetilde{P} = \widetilde{F} \circ \mathtt{Tr}_{m_0}$ and $\widetilde{P'} = \widetilde{F'} \circ \mathtt{Tr}_{m_0}$; we have that $\widetilde{\mathbf{P}}[\widetilde{F}] \approx \widetilde{P}$ and $\widetilde{\mathbf{P}}[\widetilde{F'}] \approx \widetilde{P'}$, and, by definition, $\mathtt{red}_\omega(\widetilde{\mathbf{P}}) \subset \mathtt{red}_\omega(\widetilde{\mathbf{F}})$.

We prove that the relation

$$\mathcal{R} \triangleq \{(S,T) \mid \exists \mathbf{P}', \text{ s.t. } S \approx \mathbf{P}'[\widetilde{F}], T \approx \mathbf{P}'[\widetilde{F'}] \text{ and } \mathbf{P}' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})\}$$

is a bisimulation relation satisfying $\widetilde{P} \mathcal{R} \widetilde{Q}$.

We consider $(S,T) \in \mathcal{R}$, that is, $S \approx \mathbf{P}'[\widetilde{F}]$ and $T \approx \mathbf{P}'[\widetilde{F}]'$, for some $\mathbf{P}' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$. We reason about the possible transitions from S.

1. Suppose $S \xrightarrow{\tau} S'$. We deduce:

- $\mathbf{P}'[\widetilde{F}] \Longrightarrow S'' \approx S'$ (by bisimilarity).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \Longrightarrow \mathbf{P}''$ and $\mathbf{P}''[\widetilde{F}] \approx S''$ for some n, $\mathbf{P}''$ ($\widetilde{\mathbf{F}}$ protects its solutions).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \Longrightarrow \mathbf{P}''[\widetilde{F}']$ (by Lemma 2.11(1)).
- $T \approx \mathbf{P}'[\widetilde{F}'] \approx \mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']]$ ($T \approx \mathbf{P}'[\widetilde{F}']$ and $\widetilde{F}' \approx \widetilde{\mathbf{F}}^n[\widetilde{F}']$, because $\widetilde{F}'$ is a solution of $\widetilde{\mathbf{F}}$).
- $T \Longrightarrow T' \approx \mathbf{P}''[\widetilde{F}']$ (by bisimilarity, from $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \Longrightarrow \mathbf{P}''[\widetilde{F}']$).

This situation can be depicted on the following diagram:

$$\begin{array}{ccccccccccc}
S & \approx & \mathbf{P}'[\widetilde{F}] & \approx & \mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}]] & & \mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] & \approx & \mathbf{P}'[\widetilde{F}'] & \approx & T \\
{\scriptstyle\tau}\downarrow & & \Downarrow & & \Downarrow & = & \Downarrow & & & & \Downarrow \\
S' & \approx & S'' & \approx & \mathbf{P}''[\widetilde{F}] & & \mathbf{P}''[\widetilde{F}'] & & \approx & & T'
\end{array}$$

From $\mathbf{P}' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$ and $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \Longrightarrow \mathbf{P}''$, we deduce that $\mathbf{P}'' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$. Finally, $T' \approx \mathbf{P}''[\widetilde{F}']$ and $S' \approx \mathbf{P}''[\widetilde{F}]$ give that $(S, T) \in \mathcal{R}$.

2. Similarly, suppose $S \xrightarrow{\ell} S'$. We deduce:

- $\mathbf{P}'[\widetilde{F}] \stackrel{\ell}{\Rightarrow} S'' \approx S'$ (by bisimilarity).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{\ell}{\Rightarrow} \mathbf{P}''$ and $\mathbf{P}''[\widetilde{F}] \approx S''$ for some n, $\mathbf{P}''$ ($\widetilde{\mathbf{F}}$ protects its solutions).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \stackrel{\ell}{\Rightarrow} \mathbf{P}''[\widetilde{F}']$ (by Lemma 2.11(1))
- $T \approx \mathbf{P}'[\widetilde{F}'] \approx \mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']]$ ($T \approx \mathbf{P}'[\widetilde{F}']$ and $\widetilde{F}' \approx \widetilde{\mathbf{F}}^n[\widetilde{F}']$, because $\widetilde{F}'$ is a solution of $\widetilde{\mathbf{F}}$).
- $T \stackrel{\ell}{\Rightarrow} T' \approx \mathbf{P}''[\widetilde{F}']$ (by bisimilarity, from $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \stackrel{\ell}{\Rightarrow} \mathbf{P}''[\widetilde{F}']$).

From $\mathbf{P}' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$ and $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{\ell}{\Rightarrow} \mathbf{P}''$, we deduce that $\mathbf{P}'' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$. Finally, $T' \approx \mathbf{P}''[\widetilde{F}']$ and $S' \approx \mathbf{P}''[\widetilde{F}]$ give that $(S, T) \in \mathcal{R}$.

3. Suppose now $S \stackrel{a}{\Longrightarrow} F$. We deduce:

- $\mathbf{P}'[\widetilde{F}] \stackrel{a}{\Longrightarrow} F'$ and $F \circ \mathtt{Tr}_m \approx F' \circ \mathtt{Tr}_m$ for some m (by bisimilarity).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{a}{\Longrightarrow} \mathbf{F}$ and $(\mathbf{F} \circ \mathtt{Tr}_m)[\widetilde{F}] \approx F' \circ \mathtt{Tr}_m$ for some n, $\mathbf{F}$ ($\widetilde{\mathbf{F}}$ protects its solutions).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \stackrel{a}{\Longrightarrow} \mathbf{F}[\widetilde{F}']$. (by Lemma 2.11(1)).
- $T \approx \mathbf{P}'[\widetilde{F}'] \approx \mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']]$ ($T \approx \mathbf{P}'[\widetilde{F}']$ and $\widetilde{F}' \approx \widetilde{\mathbf{F}}^n[\widetilde{F}']$, because $\widetilde{F}'$ is a solution of $\widetilde{\mathbf{F}}$).
- $T \stackrel{a}{\Longrightarrow} F''$ and $F'' \circ \mathtt{Tr}_m \approx \mathbf{F}[\widetilde{F}'] \circ \mathtt{Tr}_m = (\mathbf{F} \circ \mathtt{Tr}_m)[\widetilde{F}']$ for some F'' (by bisimilarity, from $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \stackrel{a}{\Longrightarrow} \mathbf{F}[\widetilde{F}']$).

From $\mathbf{P}' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$ and $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{a}{\Longrightarrow} \mathbf{F}$, we deduce that $\mathbf{F} \circ \mathtt{Tr}_m \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$. Finally, $F'' \circ \mathtt{Tr}_m \approx (\mathbf{F} \circ \mathtt{Tr}_m)[\widetilde{F}']$ and $F \circ \mathtt{Tr}_m \approx (\mathbf{F} \circ \mathtt{Tr}_m)[\widetilde{F}]$ give that $(F \circ \mathtt{Tr}_m, F'' \circ \mathtt{Tr}_m) \in \mathcal{R}$.

4. Again, suppose now that $S \stackrel{\overline{a}}{\Rightarrow} C$. Again:

- $\mathbf{P}'[\widetilde{F}] \stackrel{\overline{a}}{\Rightarrow} C'$ and $C \bullet \mathtt{Ab}_m \approx C' \bullet \mathtt{Ab}_m$ for some m (by bisimilarity).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{\overline{a}}{\Rightarrow} \mathbf{C}$ and $(\mathbf{C} \bullet \mathtt{Ab}_m)[\widetilde{F}] \approx C' \bullet \mathtt{Ab}_m$ for some n, $\mathbf{C}$ ($\widetilde{\mathbf{F}}$ protects its solutions).
- $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \stackrel{\overline{a}}{\Rightarrow} \mathbf{C}[\widetilde{F}']$ (by Lemma 2.11(1)).
- $T \approx \mathbf{P}'[\widetilde{F}'] \approx \mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']]$ ($T \approx \mathbf{P}'[\widetilde{F}'] \approx \mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']]$, because $\widetilde{F}'$ is a solution of $\widetilde{\mathbf{F}}$).

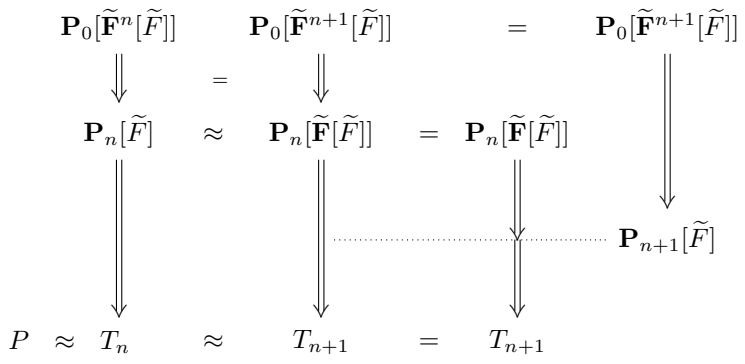

Figure 1: Recursion: construction of the sequence of transitions $\mathbf{P}_0[\mathbf{F}^n] \Longrightarrow \mathbf{P}_n$

- $T \stackrel{\overline{a}}{\Rightarrow} C''$ and $C'' \bullet \mathtt{Ab}_m \approx \mathbf{C}[\widetilde{F}'] \bullet \mathtt{Ab}_m = (\mathbf{C} \bullet \mathtt{Ab}_m)[\widetilde{F}']$ (by Lemma 2.12 and by bisimilarity, from $\mathbf{P}'[\widetilde{\mathbf{F}}^n[\widetilde{F}']] \stackrel{a}{\Longrightarrow} \mathbf{F}[\widetilde{F}']$).

  From $\mathbf{P}' \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$ and $\mathbf{P}'[\widetilde{\mathbf{F}}^n] \stackrel{\overline{a}}{\Rightarrow} \mathbf{C}$, we get that $\mathbf{C} \bullet \mathtt{Ab}_m \in \mathtt{red}_\omega(\widetilde{\mathbf{P}})$.

  Finally, $C'' \bullet \mathtt{Ab}_m \approx (\mathbf{C} \bullet \mathtt{Ab}_m)[\widetilde{F}']$ and $C \bullet \mathtt{Ab}_m \approx (\mathbf{C} \bullet \mathtt{Ab}_m)[\widetilde{F}]$ allow us to deduce that $(C \bullet \mathtt{Ab}_m, C'' \bullet \mathtt{Ab}_m) \in \mathcal{R}$.

We reason symmetrically for the actions of T. □

We say that the syntactic solutions of the system $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$ do not diverge if, for all $i \in I$, $K_{\widetilde{\mathbf{F}},i}$ does not diverge.

Theorem 3.3 (Unique solution)**.** *A guarded system of equations whose syntactic solutions do not diverge has a unique solution for* $\approx$*.*

Proof. Given a guarded system of equations $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$ whose syntactic solutions do not diverge, we prove that $\widetilde{\mathbf{F}}$ protects its solutions.

To prove that, we need to consider some $\mathbf{P}_0 \in \mathtt{red}_\omega(\widetilde{\mathbf{F}})$ and some solution $\widetilde{F}$ of $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$. There are four cases to consider, according to the four clauses of Definition 3.1.

1. Consider a transition $\mathbf{P}_0[\widetilde{F}] \Longrightarrow P$.

   We build a sequence of extended processes $\mathbf{P}_n$ and an increasing sequence of transitions $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \Longrightarrow \mathbf{P}_n$ such that: either this construction stops, yielding a transition $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \Longrightarrow \mathbf{P}_n$, or the construction is infinite, therefore giving a divergence of $\widetilde{K}_{\widetilde{\mathbf{F}}}$.

   We build this sequence so that it additionally satisfies:

   - $\mathbf{P}_n[\widetilde{F}] \Longrightarrow\approx P$,
   - The sequence is strictly increasing: the sequence of transitions $\mathbf{P}_0[\widetilde{\mathbf{F}}^{n+1}] \Longrightarrow \mathbf{P}_n[\widetilde{\mathbf{F}}]$ is a strict prefix of the sequence of transitions $\mathbf{P}_0[\widetilde{\mathbf{F}}^{n+1}] \Longrightarrow \mathbf{P}_{n+1}$ (hence $\mathbf{P}_n[\widetilde{\mathbf{F}}] \Longrightarrow \mathbf{P}_{n+1}$).

   This construction is illustrated by Figure 2. We initialise with the empty sequence from $\mathbf{P}_0$.

   Then, at step n, we have $\mathbf{P}_0[\widetilde{\mathbf{F}}^n[\widetilde{F}]] \Longrightarrow \mathbf{P}_n[\widetilde{F}] \Longrightarrow T_n$, with $P \approx T_n$.

   - If $\mathbf{P}_n[\widetilde{F}] \Longrightarrow T_n$ is the empty sequence, we stop. We have in this case $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \Longrightarrow \mathbf{P}_n$ and $P \approx \mathbf{P}_n[\widetilde{F}]$.
   - Otherwise (as depicted on Figure 1), we unfold the equations in $\widetilde{\mathbf{F}}$ one more time. We deduce

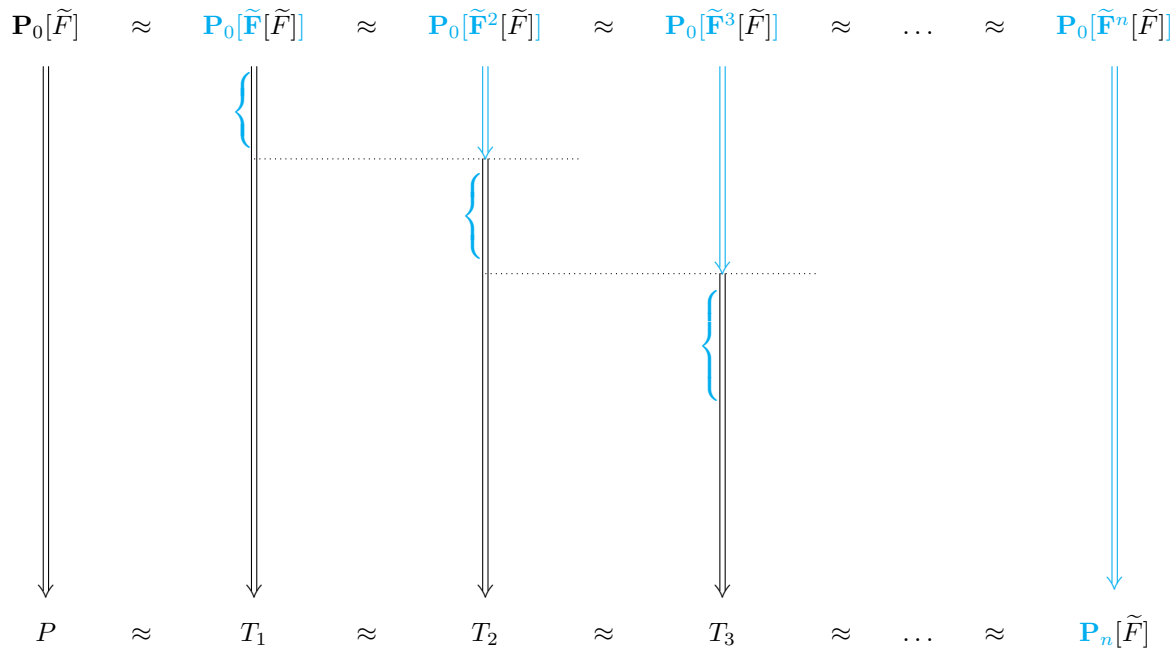

Figure 2: Construction of the transition

- $\mathbf{P}_0[\widetilde{\mathbf{F}}^{n+1}] \Longrightarrow \mathbf{P}_n[\widetilde{\mathbf{F}}]$ (by Lemma 2.11), and
- $\mathbf{P}_n[\widetilde{\mathbf{F}}[F]] \Longrightarrow T_{n+1} \approx P$ for some T_{n+1} (by congruence and bisimilarity).

If $\mathbf{P}_n[\widetilde{\mathbf{F}}[F]] \Longrightarrow T_{n+1}$ is the empty sequence, we stop as previously. Otherwise, we take $\mathbf{P}_n[\widetilde{\mathbf{F}}] \Longrightarrow \mathbf{P}_{n+1}$ to be the longest prefix sequence of transitions in $\mathbf{P}_n[\widetilde{\mathbf{F}}[F]] \Longrightarrow T_{n+1}$ that are instances of transitions from $\mathbf{P}_n[\widetilde{\mathbf{F}}]$ (we remark that as $\mathbf{P}_n[\widetilde{\mathbf{F}}]$ is guarded, this sequence is not empty).

Suppose now that the construction given above never stops. We know that $\mathbf{P}_n[\widetilde{\mathbf{F}}] \Longrightarrow \mathbf{P}_{n+1}$, therefore $\mathbf{P}_n[\widetilde{K}_{\widetilde{\mathbf{F}}}] \Longrightarrow \mathbf{P}_{n+1}[\widetilde{K}_{\widetilde{\mathbf{F}}}]$. This gives an infinite sequence of transitions starting from $\mathbf{P}_0[\widetilde{K}_{\widetilde{\mathbf{F}}}]$: $\mathbf{P}_0[\widetilde{K}_{\widetilde{\mathbf{F}}}] \Longrightarrow \mathbf{P}_1[\widetilde{K}_{\widetilde{\mathbf{F}}}] \Longrightarrow \ldots$. We observe that in the latter sequence, every step involves at least one transition, and moreover, there is no visible action occurring in this infinite sequence. Therefore $\mathbf{P}_0[\widetilde{K}_{\widetilde{\mathbf{F}}}]$ is divergent, which contradicts the hypothesis of the theorem. Hence, the construction does stop, and this concludes the proof.

Before moving on to the next case, we can remark that here, in contrast with the situation in CCS, no τ can follow an action triggered by a prefix in a weak transition: by definition (Section 1.2), $\stackrel{\mu}{\Longrightarrow}$ stands for $\Longrightarrow$ followed by $\stackrel{\mu}{\longrightarrow}$. This means that in the construction given above, (i) we stop as soon as said action is performed, and (ii) we directly build a divergence if the construction never stops, rather than a sequence of transitions leading to a divergence.

2. Now consider a transition $\mathbf{P}_0[\widetilde{F}] \stackrel{\ell}{\Rightarrow} P$. The construction is similar to that of the previous point, for a transition $\Longrightarrow$; we just build the sequence $\mathbf{P}_n$ so that it satisfies $\mathbf{P}_n[\widetilde{F}] \stackrel{\ell}{\Rightarrow}\approx P$.

   For the same reason as in the previous case, the construction necessarily stops, otherwise we would observe a divergence of the syntactic solutions.

3. Consider a transition $\mathbf{P}_0[\widetilde{F}] \stackrel{a}{\Rightarrow} F'$. Likewise, we build an increasing sequence of transitions $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \Longrightarrow$

$\mathbf{P}_n$, such that when the construction stops, it yields a transition $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \stackrel{a}{\Longrightarrow} \mathbf{F}'$, for some equation abstraction $\mathbf{F}'$ such that $(\mathbf{F}' \circ \mathtt{Tr}_m)[\widetilde{F}] \approx F' \circ \mathtt{Tr}_m$, for some m fresh. We fix such an m.

The sequence also satisfies:

- $\mathbf{P}_n[\widetilde{F}] \stackrel{a}{\Longrightarrow} G_n$ for some G_n, and $G_n \circ \mathtt{Tr}_m \approx (\mathbf{F}' \circ \mathtt{Tr}_m)[\widetilde{F}]$.
- The sequence is strictly increasing.

Again, we initialise with the empty sequence from $\mathbf{P}_0$. Suppose we are at step n. We then have

- $\mathbf{P}_0[\mathbf{F}^n[\widetilde{F}]] \Longrightarrow \mathbf{P}_n[\widetilde{F}] \stackrel{a}{\Longrightarrow} G_n$ with $G_n \circ \mathtt{Tr}_m \approx F' \circ \mathtt{Tr}_m$.
- $\mathbf{P}_0[\widetilde{\mathbf{F}}^{n+1}] \Longrightarrow \mathbf{P}_n[\widetilde{\mathbf{F}}]$ (by Lemma 2.11).
- $\mathbf{P}_n[\widetilde{\mathbf{F}}[\widetilde{F}]] \stackrel{a}{\Longrightarrow} G_{n+1}$ for some G_{n+1} such that $G_{n+1} \circ \mathtt{Tr}_m \approx G_n \circ \mathtt{Tr}_m$ (by congruence and bisimilarity).

If $\mathbf{P}_n[\widetilde{\mathbf{F}}[\widetilde{F}]] \stackrel{a}{\Longrightarrow} G_{n+1}$ is an instance of a transition from $\mathbf{P}_n[\widetilde{\mathbf{F}}]$, meaning $\mathbf{P}_n[\widetilde{\mathbf{F}}] \stackrel{a}{\Longrightarrow} \mathbf{F}'$ for some $\mathbf{F}'$ such that $\mathbf{F}'[\widetilde{F}] = G_{n+1}$, we stop. Such a transition $\mathbf{P}_n[\widetilde{\mathbf{F}}] \stackrel{a}{\Longrightarrow} \mathbf{F}'$ satisfies indeed the desired properties.

Otherwise, we take $\mathbf{P}_n[\widetilde{\mathbf{F}}] \Longrightarrow \mathbf{P}_{n+1}$ to be the longest prefix sequence of transitions in $\mathbf{P}_n[\widetilde{\mathbf{F}}[\widetilde{F}]] \stackrel{a}{\Longrightarrow} G_{n+1}$ that are instances of transitions from $\mathbf{P}_n[\widetilde{\mathbf{F}}]$. It must be a strict prefix, otherwise $\mathbf{P}_n[\widetilde{\mathbf{F}}[\widetilde{F}]] \stackrel{a}{\Longrightarrow} G_{n+1}$ would be an instance of a transition from $\mathbf{P}_n[\widetilde{\mathbf{F}}]$; it is indeed a transition $\Longrightarrow$ (as the input a must be the last performed action), and since $\mathbf{P}_n[\widetilde{\mathbf{F}}]$ is guarded, it is not the empty sequence.

Again, the construction necessarily stops, otherwise it gives an infinite sequence of transitions from $\mathbf{P}_0[\widetilde{K}_{\widetilde{\mathbf{F}}}]$, hence a divergence of the syntactic solutions.

4. Consider a transition $\mathbf{P}_0[\widetilde{F}] \stackrel{\overline{a}}{\Rightarrow} C$. The proof is similar to the previous case: we build an increasing sequence of transitions $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \Longrightarrow \mathbf{P}_n$; when the construction stops, it yields a transition $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \stackrel{\overline{a}}{\Rightarrow} \mathbf{C}$ such that $(\mathbf{C} \bullet \mathtt{Ab}_m)[\widetilde{F}] \approx C \bullet \mathtt{Ab}_m$ (m fixed). The sequence is still strictly increasing, and $\mathbf{P}_n[\widetilde{F}] \stackrel{\overline{a}}{\Rightarrow} C_n$ with $C_n \bullet \mathtt{Ab}_m \approx (\mathbf{C} \bullet \mathtt{Ab}_m)[\widetilde{F}]$.

   At step n, we have

   - $\mathbf{P}_0[\widetilde{\mathbf{F}}^n[\widetilde{F}]] \Longrightarrow \mathbf{P}_n[\widetilde{F}] \stackrel{\overline{a}}{\Rightarrow} C_n$ with $C_n \bullet \mathtt{Ab}_m \approx C \bullet \mathtt{Ab}_m$.
   - $\mathbf{P}_0[\widetilde{\mathbf{F}}^{n+1}] \Longrightarrow \mathbf{P}_n[\widetilde{\mathbf{F}}]$.
   - $\mathbf{P}_n[\widetilde{\mathbf{F}}[\widetilde{F}]] \stackrel{\overline{a}}{\Rightarrow} C_{n+1}$ for C_{n+1} such that $C_{n+1} \bullet \mathtt{Ab}_m \approx C_n \bullet \mathtt{Ab}_m$.

   If $\mathbf{P}_n[\widetilde{\mathbf{F}}[\widetilde{F}]] \stackrel{\overline{a}}{\Rightarrow} C_{n+1}$ is an instance of a transition from $\mathbf{P}_n[\widetilde{\mathbf{F}}]$ we stop. Otherwise, we take $\mathbf{P}_n[\widetilde{\mathbf{F}}] \Longrightarrow \mathbf{P}_{n+1}$ to be the longest strict prefix sequence of transitions in $\mathbf{P}_n[\widetilde{\mathbf{F}}[\widetilde{F}]] \stackrel{a}{\Longrightarrow} C_{n+1}$ that are instances transitions from $\mathbf{P}_n[\widetilde{\mathbf{F}}]$. It is not the empty sequence.

   Again, the termination argument still applies.

This concludes the proof. □

4 Innocuous Divergences

We show in this section that we can ignore some divergences in the conditions of Theorem 3.3, namely the divergences that show up in a finite unfolding of the equation (i.e., observing such divergences does not require the infinite unfolding).

Definition 4.1 (Innocuous divergence)**.** *Consider a guarded system of equations* $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$ *and its syntactic solutions* $\widetilde{K}_{\widetilde{\mathbf{F}}}$*, and suppose that* $K_{\widetilde{\mathbf{F}},i}$ *exhibits a divergence for some* i*.*

The divergence is given by a sequence of processes $(Q_k)_{k\geq 0}$ *such that* $Q_0 \in \mathtt{red}(K_{\widetilde{\mathbf{F}},i})$ *and, for all* $k \geq 0$*,* $Q_k \xrightarrow{\tau} Q_{k+1}$*.*

If there exists $n \geq 0$ *such that for all* $k \geq n$*, the derivation of the transition* $Q_k \xrightarrow{\tau} Q_{k+1}$ *does not use the transition rule for constants (Table 1) with any of the syntactic solution* $K_{\widetilde{\mathbf{F}},j}$*, we say that the divergence is* innocuous*.*

Theorem 4.2 (Unique solution with innocuous divergences)**.** *Let* $\widetilde{\mathbf{X}} = \widetilde{\mathbf{F}}$ *be a system of guarded equations, and* $\widetilde{K}_{\widetilde{\mathbf{F}}}$ *be its syntactic solutions. If for any* i*, all divergences of* $K_{\widetilde{\mathbf{F}},i}$ *are innocuous, then* $\widetilde{\mathbf{F}}$ *has a unique solution for* $\approx$*.*

Proof. We reason like in the proof of Theorem 3.3. Cases 2, 3 and 4, corresponding to inputs and outputs, are handled in the same way in the present proof.

The differences arise in case 1, corresponding to a τ transition.

- If at some point the transition $\mathbf{P}_0[\widetilde{\mathbf{F}}^n[\widetilde{F}]] \stackrel{\tau}{\Rightarrow} T_n$ is an instance of a transition $\mathbf{P}_0[\widetilde{\mathbf{F}}^n] \stackrel{\tau}{\Rightarrow} \mathbf{P}_n$, then the construction can stop.
- If on the other hand the construction never stops, we can build a non innocuous divergence: we can assume that for any n, $\mathbf{P}_0[\widetilde{\mathbf{F}}^n[\widetilde{F}]] \stackrel{\tau}{\Rightarrow} T_n$ is not an instance of a transition from $\mathbf{P}_0[\widetilde{\mathbf{F}}^n]$ (case 1), and likewise for transitions $\mathbf{P}_0[\widetilde{\mathbf{F}}^n[\widetilde{F}]] \stackrel{a}{\Rightarrow} F_n$ or $\mathbf{P}_0[\widetilde{\mathbf{F}}^n[\widetilde{F}]] \stackrel{\overline{a}}{\Rightarrow} C_n$. This means that in the sequence of transitions leading to the divergence, the constant rule is used at least n times applied to the constants $K_{\widetilde{\mathbf{F}},i}$ for various i. Hence, the divergence we build uses at least n times the constant rule for all n. Therefore, it is not innocuous, which is a contradiction.

□

An example of an innocuous divergence is when the body of an equation $\mathbf{P}$ is able to output a certain process P. In this case, the syntactic solution of $\mathbf{P}$ inherits the divergences of P, which are innocuous.

Example 4.3. *Consider the equation*

$$\mathbf{X} = \overline{a}.\langle P\rangle\mathbf{P}$$

where P *diverges (take for instance* $P = !\tau$*). The syntactic solution of this equation diverges: indeed we have (using* $\rightarrowtriangle$ *from Definition 2.13):*

$$\overline{a}.\langle P\rangle\mathbf{P} \rightarrowtriangle !m.\, P \mid \mathbf{P}\ .$$

After an input on m*,* P *is active, and the divergence can happen. However this divergence is innocuous, and does not prevent unique solution, as there is no need to unfold the equation more than once to unleash the divergence.*

We now show an example of a non-innocuous divergence that does not prevent the equation from having a unique solution.

Example 4.4. *Consider the single equation*

$$\mathbf{X} = \overline{a}.\langle !\,\overline{n}\rangle \ \mid\ n.\mathbf{X}\ .$$

Its syntactic solution, which is given by $K \stackrel{\triangle}{=} \overline{a}.\langle !\,\overline{n}\rangle \mid n.K$*, has a divergence. Indeed we have (again, using* $\rightarrowtriangle$ *from Definition 2.13):*

$$\begin{array}{rcl} K & \rightarrowtriangle & !m.!\overline{n} \mid n.K \quad \textit{for } m \textit{ fresh (i.e., } m \neq n\textit{)} \\ & \rightarrowtriangle & !m.!\overline{n} \mid !\overline{n} \mid n.K \end{array}$$

and the latter process can do an infinite sequence of τ-transitions, through synchronisations on channel n. This divergence involves the unfolding of K after each τ-step, so it is not innocuous.

However, the equation has a unique solution, intuitively because the transition at $\overline{a}$, which unleashes the divergence, can be avoided; therefore such a divergence causes no harm. We see here, therefore, a limitation of the notion of innocuous divergence.

5 An example

In this section we show an application of the unique-solution technique discussed in the paper. We consider the proof of the equality between the terms $P_1 \stackrel{\text{def}}{=} \overline{a}\,.\,\langle b\,.\,(Y)\,P\rangle$ and $Q_1 \stackrel{\text{def}}{=} \overline{a}\,.\,\langle b\,.\,(Y)\,Q\rangle$, where

$$\begin{array}{rcl} P & \stackrel{\text{def}}{=} & \boldsymbol{\nu} a\,(d.\,R \mid \overline{a}\,.\,\langle R\rangle) \\ R & \stackrel{\text{def}}{=} & a\,.\,(X)\,(Y \mid d.\,X \mid \overline{a}\,.\,\langle X\rangle) \end{array}$$

and

$$\begin{array}{rcl} Q & \stackrel{\text{def}}{=} & \boldsymbol{\nu} a\,(S \mid \overline{a}\,.\,\langle Y \mid S\rangle) \\ S & \stackrel{\text{def}}{=} & a\,.\,(X)\,(d.\,X \mid \overline{a}\,.\,\langle X\rangle) \end{array}$$

Terms P_1 and Q_1 send on a terms that can receive a process at b and then make this process freely available, at a channel d (i.e., arbitrarily many copies of the process may be activated, using d). Essentially, processes P and Q correspond to two ways of modelling a replication operator in HOπ, with a different internal structure. Indeed, if T is the process received at b, and that replaces Y in P and Q, with the abbreviations $P_T \stackrel{\text{def}}{=} P\{T/Y\}$ and $R_T \stackrel{\text{def}}{=} R\{T/Y\}$, we have

$$\begin{array}{rcl} P_T & \stackrel{d}{\longrightarrow} & \boldsymbol{\nu} a\,(R_T \mid \overline{a}\,.\,\langle R_T\rangle) \\ & \stackrel{\tau}{\longrightarrow} & \boldsymbol{\nu} a\,(T \mid d.\,R_T \mid \overline{a}\,.\,\langle R_T\rangle) \\ & \equiv & T \mid P_T \\ & \stackrel{d}{\longrightarrow} & \ldots \end{array}$$

where $\equiv$ indicates the application of a simple algebraic law for shrinking the scope of a restriction. Similarly, for $Q_T \stackrel{\text{def}}{=} Q\{T/Y\}$ and $S_T \stackrel{\text{def}}{=} S\{T/Y\}$ we have

$$\begin{array}{rcl} Q_T & \stackrel{\tau}{\longrightarrow} & \boldsymbol{\nu} a\,(d.\,(T \mid S_T) \mid \overline{a}\,.\,\langle T \mid S_T\rangle) \\ & \stackrel{d}{\longrightarrow} & \boldsymbol{\nu} a\,(T \mid S_T \mid \overline{a}\,.\,\langle T \mid S_T\rangle) \\ & \equiv & T \mid \boldsymbol{\nu} a\,(S_T \mid \overline{a}\,.\,\langle T \mid S_T\rangle) \\ & \stackrel{\tau}{\longrightarrow} & T \mid \boldsymbol{\nu} a\,(d.\,(T \mid S_T) \mid \overline{a}\,.\,\langle T \mid S_T\rangle) \\ & \stackrel{d}{\longrightarrow} & \ldots \end{array}$$

We prove $P_1 \approx Q_1$ using the technique of unique solution of equations. We use the following equations

$$\begin{array}{rcl} \mathbf{X}_1 & = & \overline{a}\,.\,\langle b\,.\,(Y)\,(\mathbf{X}_2 \circ Y)\rangle \\ \mathbf{X}_2 & = & (Y)\,d.\,(Y \mid (\mathbf{X}_2 \circ Y)) \end{array}$$

The equations are guarded and their unfoldings do not introduce divergences, hence we can apply Theorem 3.3. We derive $P_1 \approx Q_1$ by showing that both the pair P_1, $(Y)\,P$ and the pair Q_1, $(Y)\,Q$ are solutions of the system. For the first pair we have to prove

$$\begin{array}{rcl} P_1 & \approx & \overline{a}\,.\,\langle b\,.\,(Y)\,P\rangle \\ (Y)\,P & \approx & (Y)\,d.\,(Y \mid P) \end{array}$$

The first equivalence is trivial, as the two processes are identical. For the second, by definition of normal bisimulation, for some m fresh, and writing P_m as abbreviation for $P\{\overline{m}.\,\mathbf{0}/Y\}$, we have to show

$$P_m \approx d.\,(\overline{m}.\,\mathbf{0} \mid P_m)$$

We have, for $R_m \stackrel{\text{def}}{=} R\{\overline{m}.\,\mathbf{0}/Y\}$:

$$\begin{array}{lcl} P_m & = & \boldsymbol{\nu} a\,(d.\,R_m \mid \overline{a}\,.\,\langle R_m\rangle) \\ & \approx & d.\,\boldsymbol{\nu} a\,(R_m \mid \overline{a}\,.\,\langle R_m\rangle) \\ & = & d.\,\boldsymbol{\nu} a\,(a\,.\,(X)\,(\overline{m}.\,\mathbf{0} \mid d.\,X \mid \overline{a}\,.\,\langle X\rangle) \mid \overline{a}\,.\,\langle R_m\rangle) \\ & \approx & d.\,\boldsymbol{\nu} a\,((\overline{m}.\,\mathbf{0} \mid d.\,R_m \mid \overline{a}\,.\,\langle R_m\rangle)) \\ & \approx & d.\,(\overline{m}.\,\mathbf{0} \mid \boldsymbol{\nu} a\,((d.\,R_m \mid \overline{a}\,.\,\langle R_m\rangle))) \\ & = & d.\,(\overline{m}.\,\mathbf{0} \mid P_m) \end{array}$$

where the uses of $\approx$ are derived using some simple algebraic laws for prefix and restriction plus (for the second occurrence of $\approx$) the law

$$\begin{array}{lcl} \boldsymbol{\nu} a\,(a\,.\,(X)\,T_1 \mid \overline{a}\,.\,\langle T_2\rangle T_3) & \approx & \boldsymbol{\nu} a\,(((X)\,T_1) \bullet (\langle T_2\rangle T_3)) \\ & = & \boldsymbol{\nu} a\,(T_1\{^{T_2}\!/_X\} \mid T_3) \end{array}$$

which is straightforward too, as the process on the right is the only immediate derivative of the process on the left. The reasoning for the other pair, Q_1, $(Y)\,Q$, is similar (in fact, simpler).

The proof above makes use of two equations only. One may find this surprising, because the calculus is higher-order and therefore process terms are exchanged with the environment (i.e., the input at b and the output at a), and because of the recursive structure of the terms compared (their 'fixed-point-like' behaviour). Such reduced size is due both to the choice of normal bisimulation (that does not make use of universal quantifications on concretions or abstractions in the input and output clauses), and to the 'up-to context' flavour of the unique-solution technique.

When comparing the technique of unique solution of equations to the bisimulation proof method [San15] the size of a system of equations is the number of equations involved, whereas the size of a bisimulation is the number of its pairs. Thus the proof above, involving two equations, would correspond to a bisimulation candidate with two pairs. The proof of a similar result in [SKS11] (Section 6.7, second example) uses 'environmental bisimulation up-to context'. The relation employed in that proof is infinite, because in environmental bisimulation one has to take into account all possible processes that may be received in a higher-order input and because of the limitations of the known forms of 'up-to context' for environmental bisimulation (only in certain clauses of the bisimulation a context may be erased and even in these cases there are syntactic constraints on the context itself).

6 Conclusion

In the paper we have proposed a new bisimulation technique for the Higher-Order π-calculus (HOπ), based on unique solution of equations. The technique implicitly enables 'up-to context' reasoning, something known to be important in higher-order languages. As bisimilarity we have focused on normal bisimilarity, both for its efficiency (its clauses do not require universal quantifications on terms provided by the observer), and because direct proofs of forms of 'bisimulation up-to context' for it appear challenging.

We have focused on the monadic calculus and on monadic abstractions, and considered only process passing. The extension to polyadicity and to the communication of abstractions (i.e., parametrised processes) of arbitrarily high type, as in the full HOπ [San92], appears to be only notationally more complex. However, handling more sophisticated types for the terms exchanged (i.e., abstractions with recursive types or polymorphism) would seem challenging.

We believe that Theorems 3.3 and 4.2 can be adapted to other (weak) behavioural equivalences and preorders, as is the case of their counterpart in CCS [DHS17]. However, with context bisimulation (discussed in Section 1.2), the technique and the treatment of divergence might need refinements, because, as the clauses of the bisimulation involve universal quantifications on terms supplied by the external environment, the current conditions on divergence might be too restrictive.

Another approach to the 'unique-solution' proof technique involves *inequations* and the *contraction* preorder, in place of equations [San15]. In the case of normal bisimilarity, however, this approach does not seem

applicable because the substitutivity problems for normal bisimilarity (that also represent the main obstacle to the development of 'up-to context' enhancements for its bisimulation method) would also show up with contraction. See the discussion at the end of Section 1.2 about the expansion preorder; the problems with the contraction preorder are similar.

An issue not tackled in this paper is completeness of the technique proposed. This issue has many facets. First, there is a completeness issue with respect to normal bisimilarity: is it possible to prove all normal bisimilarities using the unique solution technique? Stated so, the question is vacuous, since any bisimilarity $P \approx Q$ could be handled with the equation

$$\mathbf{X} = P \ .$$

A more interesting question is whether for any normal bisimulation there is a system of equations with the same size (i.e., the number of equations is not larger than the number of pairs in the relation) that would prove the same equalities. An analogous question could then be formulated with respect to 'normal bisimulation up-to context', as our unique solution technique claims to allow 'up-to context' reasoning. The answer to both questions is likely to be negative, because of the condition on divergence in the unique-solution technique. It would be interesting however to see whether a positive answer can be given by introducing appropriate syntactic restrictions, or using more refined forms of divergence as suggested by Example 4.4. (Completeness holds in CCS, for inequations and contractions [San15]).

An interesting use of 'up-to context' has been put forward by Biernacki et al. in the λ-calculus [BLP18]. We would like to see if similar enhancements can also be applied to higher-order processes and to the unique-solution technique.

Acknowledgement

This work has been supported by the European Research Council (ERC) under the Horizon 2020 programme (CoVeCe, grant agreement No 678157), the ANR under the programmes "Investissements d'Avenir" (ANR-11-IDEX-0007, LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon) and Elica, ANR-14-CE25-0005, and the Université Franco-Italienne under the programme Vinci.

References

[BBR10] Jos C.M. Baeten, Twan Basten, and Michel A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*. Cambridge University Press, 2010.

[BLP18] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. Proving soundness of extensional normal-form bisimilarities. *Electr. Notes Theor. Comput. Sci.*, 336:41–56, 2018.

[DHS17] Adrien Durier, Daniel Hirschkoff, and Davide Sangiorgi. Divergence and unique solution of equations. In *Proc. of CONCUR'2017*, volume 85 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[GM14] Jan Friso Groote and Mohammad Reza Mousavi. *Modeling and Analysis of Communicating Systems*. MIT Press, 2014.

[KW06] Vassileios Koutavas and Mitchell Wand. Small bisimulations for reasoning about higher-order imperative programs. In *Proc. POPL'06*, pages 141–152, 2006.

[Las98a] S. B. Lassen. *Relational Reasoning about Functions and Nondeterminism*. PhD thesis, Department of Computer Science, University of Aarhus, 1998.

[Las98b] Søren B. Lassen. Relational reasoning about contexts. In Andrew D. Gordon and Andrew M. Pitts, editors, *Higher Order Operational Techniques in Semantics*, Publications of the Newton Institute, pages 91–135. Cambridge University Press, 1998.

[Mil89] Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.

[MN05] Massimo Merro and Francesco Zappa Nardelli. Behavioral theory for mobile ambients. *J. ACM*, 52(6):961–1023, 2005.

[MPS14] Jean-Marie Madiot, Damien Pous, and Davide Sangiorgi. Bisimulations up-to: Beyond first-order transition systems. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, pages 93–108, 2014.

[PS11] Damien Pous and Davide Sangiorgi. *Advanced Topics in Bisimulation and Coinduction (D. Sangiorgi and J. Rutten editors)*, chapter Enhancements of the coinductive proof method. Cambridge University Press, 2011.

[Ros92] A. W. Roscoe. An alternative order for the failures model. *J. Log. Comput.*, 2(5):557–577, 1992.

[Ros97] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[Ros10] A. W. Roscoe. *Understanding Concurrent Systems*. Springer, 2010.

[San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh, 1992.

[San96] Davide Sangiorgi. Bisimulation for higher-order process calculi. *Inf. Comput.*, 131(2):141–178, 1996.

[San15] Davide Sangiorgi. Equations, contractions, and unique solutions. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 421–432, 2015.

[SKS11] Davide Sangiorgi, Naoki Kobayashi, and Eijiro Sumii. Environmental bisimulations for higher-order languages. *ACM Trans. Program. Lang. Syst.*, 33(1):5:1–5:69, 2011.

[SW01] Davide Sangiorgi and David Walker. *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press, 2001.