Divergence and unique solution of equations

Adrien Durier[1,2], Daniel Hirschkoff[1], and Davide Sangiorgi[2]

1 **Université de Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France**
2 **Università di Bologna and INRIA, Italy**

Abstract

We study proof techniques for bisimilarity based on *unique solution of equations*. We draw inspiration from a result by Roscoe in the denotational setting of CSP and for failure semantics, essentially stating that an equation (or a system of equations) whose infinite unfolding never produces a divergence has the unique-solution property. We transport this result onto the operational setting of CCS and for bisimilarity. We then exploit the operational approach to: refine the theorem, distinguishing between different forms of divergence; derive an abstract formulation of the theorems, on generic LTSs; adapt the theorems to other equivalences such as trace equivalence, and to preorders such as trace inclusion. We compare the resulting techniques to enhancements of the bisimulation proof method (the 'up-to techniques'). Finally, we study the theorems in name-passing calculi such as the asynchronous π-calculus, and revisit the completeness proof of Milner's encoding of the λ-calculus into the π-calculus for Lévy-Longo Trees.

1 Introduction

In this paper we study the technique of *unique solution of equations* for (weak) behavioural relations. We mainly focus on bisimilarity but we also consider other equivalences, such as trace equivalence, as well as preorders such as trace inclusion. Roughly, the technique consists in proving that two tuples of processes are componentwise in a given behavioural relation by establishing that they are solutions of the same system of equations.

In this work, behavioural relations, hence also bisimilarity, are meant to be *weak* because they abstract from internal moves of terms, as opposed to the *strong* relations, which make no distinctions between the internal moves and the external ones (i.e., the interactions with the environment). Weak equivalences are, practically, the most relevant ones: e.g., two equal programs may produce the same result with different numbers of evaluation steps. Further, the problems tackled in this paper only arise in the weak case.

The technique of unique solution has been proposed by Milner in the setting of CCS, and plays a prominent role in proofs of examples in his book [13]. The method is important in verification techniques and tools based on algebraic reasoning [22, 2, 10]. Not all equations have a unique solution: for instance any process trivially satisfies $X = X$. In Milner's theorem [13], the conditions for uniqueness of solutions have limitations: the equations must be 'strongly guarded and sequential', that is, 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 occurrences of other operators above the variables, such as parallel composition and restriction, in general cannot be removed), and its transport onto other languages (e.g., languages for distributed systems or higher-order languages usually do not include the sum operator, which makes the theorem essentially useless). A comparable technique, involving similar limitations, has been proposed by Hoare in his CSP book [12].

Trying to overcome such limitations, a variant of the technique, called *unique solution of contractions* has been proposed [25]. The technique is for behavioural equivalences; however the meaning of 'solution' is defined in terms of the contraction of the chosen equivalence.

Contraction is, intuitively, a preorder that conveys an idea of efficiency on processes, where efficiency is measured on the number of internal actions needed to perform a certain activity. The condition for applicability of the technique is, as for Milner's, purely syntactic: each variable in the body of an equation should be underneath a prefix. The technique has two main disadvantages: for proving an equivalence one needs also the theory of the associated contraction preorder; there may be processes for which the technique is not applicable simply because the contraction is strictly finer than the equivalence, and therefore one of the processes fails to be a solution.

In this paper we explore a different approach, inspired by results by Roscoe in CSP [21, 20], essentially stating that a guarded equation (or system of equations) whose infinite unfolding never produces a divergence has the unique-solution property. The theorem is presented, as usual in CSP, with respect to denotational semantics and failure based equivalence [5, 6]. In such a setting, where divergence is catastrophic (e.g., it is the bottom element of the domain), the theorem has an elegant and natural formulation. (Indeed, Roscoe develops a denotational model [20] in which the proof of the theorem is just a few lines.)

We draw inspiration from Roscoe's work to formulate the counterpart of these results in the operational setting of CCS and bisimilarity. In comparison with the denotational CSP proof, the operational CCS proof is more complex. The operational setting offers however a few advantages. First, we can formulate more refined versions of the theorem, in which we distinguish between different forms of divergence. (These refinements would look less natural in the denotational and trace-based setting of CSP, where any divergence causes a process to be considered undefined.) A second and more important advantage comes as a consequence of the flexibility of the operational approach: the unique-solution theorems can be tuned to other behavioural relations (both equivalences and preorders), and to other languages.

To highlight the latter aspect, we present abstract formulations of the theorems, on a generic LTS (i.e., without reference to CCS), where the body of an equation becomes a function on the states of the LTS. The CCS theorems are instances of the abstract formulations. Similarly we can derive analogous theorems for other languages. Indeed we can do so for all languages whose constructs have an operational semantics with rules in the GSOS format [3] (assuming appropriate hypotheses, among which congruence properties). In contrast, the analogous theorems fail for languages whose constructs follow the *tyft/tyxt* [11] format, due to the possibility of rules with a *lookahead*. We also consider extensions of the theorems to name-passing calculi such as the π-calculus. The abstract version of our main unique-solution theorem has been formalised using the Coq proof assistant [8].

Today, for concrete proofs of bisimilarity results, the bisimulation proof method is predominant, also thanks to enhancements of the method provided by the so called 'up-to techniques' [17]. Powerful enhancements are 'up to context', whereby in the derivatives of two terms a common context can be erased, 'up to expansion', whereby two derivatives can be rewritten using the expansion preorder, and 'up to transitivity', whereby the matching between two derivatives is made with respect to the transitive closure of the candidate relation (rather than the relation alone). Different enhancements can sometimes be combined, though care is needed to preserve soundness. One of the most powerful combinations is Pous *'up to transitivity and context'* technique, which relies on a termination hypothesis. This technique generalises 'up to expansion' and combines it with 'up to context' and 'up to transitivity'. We show that, under an additional side condition, our techniques are at least as powerful as this up-to technique: any up-to relation can be turned into a system of equations of the same size as the up-to relation and that satisfies the hypothesis of our theorems.

An important difference between unique solution of equations and up-to techniques

$$\texttt{sum}\ \frac{}{\Sigma_{i\in I}\mu_i.\,P_i \xrightarrow{\mu_i} P_i} \qquad \texttt{parL}\ \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q} \qquad \texttt{comL}\ \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\texttt{res}\ \frac{P \xrightarrow{\mu} P'}{\boldsymbol{\nu} a\, P \xrightarrow{\mu} \boldsymbol{\nu} a\, P'}\ \mu \neq a, \overline{a} \qquad \texttt{const}\ \frac{P \xrightarrow{\mu} P'}{K \xrightarrow{\mu} P'}\ \text{if } K \stackrel{\triangle}{=} P$$

Figure 1 The LTS for CCS

arises in the (asynchronous) π-calculus. In this setting, forms of bisimulation enhancements that involve 'up to context' require closure of the candidate relation under substitutions (or instantiation of the parameters of an abstraction with arbitrary values). It is an open problem whether this closure is necessary in the asynchronous π-calculus, where bisimilarity is closed under substitutions. Our unique-solution techniques are strongly reminiscent of up to context techniques (the body of an equation acts like a context that is erased in a proof using 'up to context'); yet, surprisingly, no closure under substitutions is required.

As an example of application of our techniques in the π-calculus we revisit the completeness part of the proof of full abstraction for the encoding of the λ-calculus into the π-calculus [24, 27] with respect to Levy-Longo Trees (LTs). The proof in [24, 27] uses 'up to expansion and context'. Such up-to techniques seem to be essential: without them, it would be hard even to define the bisimulation candidate. For our proof using unique-solution, there is one equation for each node of a given LT, describing the shape of such node.

Outline of the paper: Section 2 provides background about CCS and behavioural relations. We formulate our main results for CCS in Section 3, and generalise them in an abstract setting in Section 4. Section 5 shows how our results can be applied to the π-calculus.

2 Background

CCS. We assume an infinite set of *names* $a, b, \ldots$ and a set of *constant identifiers* (or simply *constants*) to write recursively defined processes. The special symbol τ does not occur in the names and in the constants. We recall the grammar of CCS:

$$P \ := \ P_1 \mid P_2 \ \mid \ \Sigma_{i\in I}\mu_i.\,P_i \ \mid \ \boldsymbol{\nu} a\, P \ \mid \ K \qquad\qquad \mu \ := \ a \ \mid \ \overline{a} \ \mid \ \tau$$

where I is a countable indexing set. We write $\mathbf{0}$ when I is empty, and $P + Q$ for binary sums. Each constant K has a definition $K \stackrel{\triangle}{=} P$. We sometimes omit trailing $\mathbf{0}$, e.g., writing $a \mid b$ for $a.\,\mathbf{0} \mid b.\,\mathbf{0}$. The operational semantics is given by means of an LTS, and is given in Figure 1 (the symmetric versions of the rules `parL` and `comL` have been omitted).

Some standard notations for transitions: $\Longrightarrow$ is the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\stackrel{\mu}{\Longrightarrow}$ is $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$ (the composition of the three relations). Moreover, $P \xrightarrow{\hat{\mu}} P'$ holds if $P \xrightarrow{\mu} P'$ or ($\mu = \tau$ and $P = P'$); similarly $P \stackrel{\widehat{\mu}}{\Longrightarrow} P'$ holds if $P \stackrel{\mu}{\Longrightarrow} P'$ or ($\mu = \tau$ and $P = P'$). Letters $\mathcal{R}, \mathcal{S}$ range over relations. We use the infix notation for relations, e.g., $P\ \mathcal{R}\ Q$ means that $(P, Q) \in \mathcal{R}$. We use a tilde to denote a tuple, with countably many elements; thus the tuple may also be infinite. All notations are extended to tuples componentwise; e.g., $\widetilde{P}\ \mathcal{R}\ \widetilde{Q}$ means that $P_i\ \mathcal{R}\ Q_i$, for each component i of the tuples $\widetilde{P}$ and $\widetilde{Q}$. We use $\stackrel{\text{def}}{=}$ for abbreviations; in contrast, $\stackrel{\triangle}{=}$ is used for the definition of constants, and $=$ for syntactic equality and for equations. We focus on *weak* behavioural equivalences, which abstract from the number of internal steps performed.

▶ **Definition 1** (Bisimilarity)**.** A relation $\mathcal{R}$ is a *bisimulation* if, whenever $P\ \mathcal{R}\ Q$, we have:

1. $P \xrightarrow{\mu} P'$ implies that there is Q' such that $Q \stackrel{\hat{\mu}}{\Longrightarrow} Q'$ and $P' \mathcal{R} Q'$;
2. the converse, on the actions from Q.

P and Q are *bisimilar*, written $P \approx Q$, if $P \mathcal{R} Q$ for some bisimulation $\mathcal{R}$.

Systems of equations. Unique solution of equations [13] 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.

We need variables to write equations. We use capital letters X, Y, Z for these variables and call them *equation variables*. The body of an equation is a CCS expression possibly containing equation variables. We use $E, E', \ldots$ to range over *equation expressions*; these are process expressions that may contain occurrences of variables; that is, the grammar for processes is extended with a production for variables.

▶ **Definition 2.** Assume that, for each i of a countable indexing set I, we have a variable X_i, and an expression E_i, possibly containing some variables. Then $\{X_i = E_i\}_{i \in I}$ (sometimes written $\widetilde{X} = \widetilde{E}$) is a *system of equations*. (There is one equation for each variable X_i.)

$E[\widetilde{P}]$ is the process resulting from E by replacing each variable X_i with the process P_i, assuming $\widetilde{P}$ and $\widetilde{X}$ have the same length. (This is syntactic replacement.) The components of $\widetilde{P}$ need not be different from each other, while this must hold for the variables $\widetilde{X}$.

▶ **Definition 3.** Suppose $\{X_i = E_i\}_{i \in I}$ is a system of equations. We say that:

- $\widetilde{P}$ is a *solution of the system of equations for* $\approx$ if for each i it holds that $P_i \approx E_i[\widetilde{P}]$.
- The system has *a unique solution for* $\approx$ if whenever $\widetilde{P}$ and $\widetilde{Q}$ are both solutions for $\approx$, then $\widetilde{P} \approx \widetilde{Q}$.

For instance, the system $X_1 = a.\, X_2$, $X_2 = b.\, X_1$ has a unique solution, whereas the equations $X = X$, or $X = \tau.\, X$, or $X = a \mid X$ do not.

A system of equations is *guarded* (resp. *strongly guarded*) if each occurrence of a variable in the body of an equation is underneath a prefix (resp. a visible prefix, i.e., different from τ).

Divergences. We consider the standard notion of divergence:

▶ **Definition 4** (Divergence)**.** A process P *diverges* if it can perform an infinite sequence of internal moves, possibly after some visible ones; i.e., there are processes P_i, $i \geq 0$, and some n, such that $P = P_0 \xrightarrow{\mu_0} P_1 \xrightarrow{\mu_1} P_2 \xrightarrow{\mu_2} \ldots$ and for all $i > n$, $\mu_i = \tau$. We call a *divergence of* P the sequence of transitions $\left(P_i \xrightarrow{\mu_i} P_{i+1}\right)_i$.

▶ **Example 5.** The process $L \stackrel{\triangle}{=} a.\, \boldsymbol{\nu} a\, (L \mid \overline{a})$ diverges, since $L \xrightarrow{a} \boldsymbol{\nu} a\, (L \mid \overline{a})$, and (leaving aside $\mathbf{0}$ and useless restrictions) $\boldsymbol{\nu} a\, (L \mid \overline{a})$ has a τ transition onto itself.

3 Main Results

3.1 Divergences and Unique Solution

This section is devoted to our main results for bisimilarity, in the case of CCS. We need to reason with the unfoldings of the given equation $X = E$: the n-th unfolding of E is E^n; thus E^1 is E, E^2 is $E[E]$, and E^{n+1} is $E^n[E]$. This operation is extended to a system of equations $\widetilde{X} = \widetilde{E}[\widetilde{X}]$ in a natural way (where E_i replaces X_i in the unfolding). The *infinite* unfolding represents the simplest and most intuitive solution to the equation. In the CCS grammar, such a solution is obtained by turning the equation into a constant definition, namely the

constant K_E with $K_E \stackrel{\triangle}{=} E[K_E]$. We call K_E the *syntactic solution of the equation*. Similarly for a system of equations, we call *syntactic solutions* the set of mutually recursive constants $\{K_{\widetilde{E},i} \stackrel{\triangle}{=} E_i[\widetilde{K}_{\widetilde{E}}]\}_i$.

▶ **Theorem 6** (Unique solution). *A guarded system of equations whose syntactic solutions do not diverge has a unique solution for* $\approx$.

We explain the schema of the proof, considering, for simplicity, a single equation $X = E$. We take a solution P of the equation and a transition $P \xrightarrow{\mu} P'$. The goal is to find an n such that $E^n[P]$ can match this transition *without the need of* P; i.e., there is E' with $E^n[P] \stackrel{\hat{\mu}}{\Longrightarrow} E'[P]$, and for any process Q also $E^n[Q] \stackrel{\hat{\mu}}{\Longrightarrow} E'[Q]$ holds.

We look for this n incrementally. If the matching transition $E^m[P] \stackrel{\hat{\mu}}{\Longrightarrow} P_m$ involves some transitions of P, then $E^m[P]$ does not work. We then consider a matching transition emanating from $E^{m+1}[P]$, which starts with the transitions in $E^m[P] \stackrel{\hat{\mu}}{\Longrightarrow} P_m$ that do not involve P. We observe that there are are at least m of these, because P is underneath at least m prefixes in $E^m[P]$.

This procedure necessarily stops: otherwise, we could build an infinite sequence of transitions involving only the unfoldings of E, and with at most one visible transition: this would yield a divergence in the syntactic solution of E.

With this construction at hand, given another solution Q of the equation, we construct a bisimulation containing the pair (P, Q). Appendix A provides a more detailed proof.

3.2 Innocuous Divergences

In the remainder of the section we refine Theorem 6 by taking into account only certain forms of divergence. To introduce the idea, consider the equation $X = a.\,X \mid K$, for $K \stackrel{\triangle}{=} \tau.\,K$: the divergences induced by K do not prevent uniqueness of the solution, as any solution P necessarily satisfies $P \approx a.\,P$. Indeed the variable of the equation is strongly guarded and a visible action has to be produced before accessing the variable. These divergences are not dangerous because they do not percolate through the infinite unfolding of the equation; in other words, it is not necessary to go to the infinite unfolding to diverge. We call such divergences *innocuous*. Formally, these divergences are derived by applying only a finite number of times rule `const` of the LTS (see Figure 1) to the constant that represents the syntactic solution of the equation.

▶ **Definition 7** (Innocuous divergence). Consider a guarded system of equations $\widetilde{X} = \widetilde{E}$ and its syntactic solutions $\widetilde{K}_{\widetilde{E}}$. A divergence of $K_{\widetilde{E},i}$ (for some i) is called *innocuous* when there exists a finite bound on the number of times rule `const` is applied to any of the constants $K_{\widetilde{E},j}$ in the sequence of derivation proofs of the transitions belonging to the divergence.

▶ **Theorem 8** (Unique solution with innocuous divergences). *Let* $\widetilde{X} = \widetilde{E}$ *be a system of guarded equations, and* $\widetilde{K}_{\widetilde{E}}$ *be its syntactic solutions. If all divergences of any* $K_{\widetilde{E},i}$ *are innocuous, then* $\widetilde{E}$ *has a unique solution for* $\approx$.

Appendix A presents the proof.

▶ Remark. The conditions for unique solution in Theorems 6 and 8 are a mixture of syntactic (guardedness) and semantic (divergence-free) conditions. A purely semantic condition can be used if rule `const` of Figure 1 is modified so that the unfolding of a constant yields a τ-transition:

$$\frac{}{K \xrightarrow{\tau} P} \text{ if } K \stackrel{\triangle}{=} P$$

Thus in the theorems the condition that the equations are guarded could be dropped. The resulting theorems would actually be more powerful because they would accept equations not all of which are guarded: it is sufficient that each equation has a finite unfolding that is guarded. For instance the system of equations $X = b \mid Y$, $Y = a.\, X$ would be accepted, although the first equation is not guarded.

The next lemma states a condition to ensure that all divergences produced by a system of equations are innocuous. This condition will be sufficient in all examples in the paper.

▶ **Lemma 9.** *In a system of equations $\widetilde{X} = \widetilde{E}$, suppose for each i there is n_i such that in $E_i^{n_i}$, each variable is underneath a visible prefix (say, a or $\overline{a}$) whose complementary prefix ($\overline{a}$ or a) does not appear in any equation. Then the system has only innocuous divergences.*

3.3 An example: lazy and eager servers

In this section we show an example of application of our technique, inspired by a similar one in [25]. The example also illustrates the relative strengths of the two unique solution theorems (Theorem 6 and Theorem 8).

For the sake of readability, we use a version of CCS with value passing; this could be translated into pure CCS [13]. In a value-passing calculus, $a(x).\, P$ is an input at a in which x is the placeholder for the value received, whereas $\overline{a}\langle n\rangle.\, P$ is an output at a of the value n; and $A\langle n\rangle$ is a parametrised constant. This example consists of two implementations of a server; this server, when interrogated by clients at a channel c, should start a certain interaction protocol with the client, after consulting an auxiliary server A at a.

We consider the two following implementations of this server: the first one, L, is 'lazy', and consults A only *after* a request from a client has been received. In contrast, the other one, E, is 'eager', and consults A *beforehand*, so then to be ready in answering a client:

$$\begin{array}{rcl} L & \triangleq & c(z).\, a(x).\, (L \mid R\langle x, z\rangle) \\ E & \triangleq & a(x).\, c(z).\, (E \mid R\langle x, z\rangle) \end{array} \qquad\qquad A\langle n\rangle \triangleq \overline{a}\langle n\rangle.\, A\langle n+1\rangle$$

Here $R\langle x, z\rangle$ represents the interaction protocol that is started with a client, and can be any process. It may use the values x and z (obtained from the client and the auxiliary server A); the interactions produced may indeed depend on the values x and z. We assume for now that $R\langle x, z\rangle$ may not use channel c and a; that is, the interaction protocol that has been spawned need not come back to the main server or to the auxiliary server. Moreover we assume R may not diverge. We want to prove that the two servers, when composed with A, yield bisimilar processes. We thus define $LS\langle n\rangle \triangleq \boldsymbol{\nu} a\,(A\langle n\rangle \mid L)$ and $ES\langle n\rangle \triangleq \boldsymbol{\nu} a\,(A\langle n\rangle \mid E)$. A proof that $LS\langle n\rangle \approx ES\langle n\rangle$ using the plain bisimulation proof method would be long and tedious, due to the differences between the lazy and the eager server, and to the fact that R is nearly an arbitrary process.

For a proof using our technique, the equations are: $\{X_n = c(z).\, (X_{n+1} \mid R\langle n, z\rangle)\}_n$. The proofs that the two servers are solutions can be carried out using a few algebraic laws: expansion law, structural laws for parallel composition and restriction, one τ-law. To apply Theorem 6, we also have to check that the equations may not produce divergences. This check is straightforward, as no silent move may be produced by interactions along c, and any two internal communications at a are separated by a visible input at c. Moreover, by assumption, the protocol R does not produce internal divergences.

If however the hypothesis that R may not diverge is lifted, then Theorem 6 is not applicable anymore, and divergences are possible. However, such divergences are innocuous:

the equation need not be unfolded an infinite number of times for the divergence to occur. We can therefore still prove the result, by appealing to the more powerful Theorem 8.

3.4 Comparison with other techniques

Milner's syntactic condition for unique solution of equations essentially allows only equations in which variables are underneath prefixes and sums. The technique is not complete [25]; for instance it cannot handle the server example of Section 3.3.

The technique of 'unique solution of contractions' [25] relies on the theory of an auxiliary preorder (contraction), needed to establish the meaning of 'solution'; and the soundness theorems in [25] use a purely syntactic condition (guarded variables). In contrast, our techniques with equations do not rely on auxiliary relations, but the soundness theorems use a semantic condition (divergence). The two techniques are incomparable. Considering the server example of Section 3.3, the contraction technique is capable of handling also the case in which the protocol R is freely allowed to make calls back to the main server, including the possibility that, in doing this, infinitely many copies of R are spawned. This possibility is disallowed for us, as it would correspond to a non-innocuous divergence. On the other hand, when using contraction, a solution is evaluated with respect to the contraction preorder, that conveys an idea of efficiency (measured against the number of silent transitions performed). Thus we can use our techniques to prove that processes $K \stackrel{\triangle}{=} \tau.a.a.K$ and $H \stackrel{\triangle}{=} a.H$ are bisimilar because they are solutions of the equation $X = a.X$; in contrast, they are not solutions of the corresponding contraction.

Up-to techniques. We compare our unique-solution techniques with one of the most powerful forms of enhancement of the bisimulation proof method, namely Pous 'up to transitivity and context' technique [18]. That technique allows us to use 'up to weak bisimilarity', 'up to transitivity', and 'up to context' techniques together. While 'up to weak bisimilarity' and 'up to transitivity' are known to be unsound techniques [17], here they are combined at the price of a termination hypothesis over a 'control relation', below written $\succ$.

We write $\mathcal{C}(\mathcal{R})$ for the context closure of a relation $\mathcal{R}$ (the set of all pairs $(C[\widetilde{P}], C[\widetilde{Q}])$ with $\widetilde{P}\ \mathcal{R}\ \widetilde{Q}$). Moreover, $\overline{\mathcal{R}}$ stands for $(\approx \cup \mathcal{C}(\mathcal{R}))$, and $\mathcal{R}^+$ for the transitive closure of $\mathcal{R}$.

▶ **Definition 10.** Let $\succ$ be a relation that is transitive, closed under contexts, and such that $\succ (\stackrel{\tau}{\longrightarrow}^+)$ terminates. A relation $\mathcal{R}$ is a *bisimulation up to* $\succ$ *and context* if, whenever $P\ \mathcal{R}\ Q$:

1. if $P \stackrel{\mu}{\longrightarrow} P'$ then $Q \stackrel{\hat{\mu}}{\Longrightarrow} Q'$ for some Q' with $P'\ (\succ \cap \overline{\mathcal{R}})^+\ \mathcal{C}(\mathcal{R})\ \approx\ Q'$;
2. the converse, on the transitions from Q.

If $\mathcal{R}$ is a relation then we can also view $\mathcal{R}$ as an ordered sequence of pairs (e.g., assuming some lexicographical ordering). Then $\mathcal{R}_i$ indicates the tuple obtained by projecting the pairs in $\mathcal{R}$ on the i-th component ($i = 1, 2$). The *size* of a relation is the number of pairs it contains. The size of a system of equations is the number of equations it consists of.

▶ **Theorem 11** (Completeness with respect to up-to techniques)**.** *Suppose $\mathcal{R}$ is a bisimulation up to $\succ$ and context. Then there is a guarded system of equations, of the same size as $\mathcal{R}$, with only innocuous divergences, admitting $\mathcal{R}_1$ and $\mathcal{R}_2$ as solutions.*

In the proof, the equations are defined by exploiting the expansion law in CCS [13]. The proof then involves a rather delicate analysis in which the termination hypothesis is used to prove that the syntactic solutions of this system have only innocuous divergences (they may indeed have divergences). We may then apply Theorem 8 (rather than Theorem 6). Appendix B provides more details.

▶ Remark. In [18], the transitive closure $(\succ \cap \overline{\mathcal{R}})^+$ in Definition 10 is actually a *reflexive* and transitive closure. We do not know if relaxing this technical condition breaks Theorem 11.

4 Abstract Formulation

4.1 Abstract LTS and Operators

In this section we propose generalisations of the unique-solution theorems. For this we introduce abstract formulations of them, which are meant to highlight their essential ingredients. When instantiated to the specific case of CCS, such abstract formulations yield the theorems in Section 3. The proofs are adapted from those of the corresponding theorems in Section 3. The results of this section, up to Theorem 15, have been formalised in Coq [8].

The abstract formulation is stated on a generic LTS, that is, a triple $\mathcal{T} = (S, \Lambda, \rightarrow)$ where: S is the set of states; Λ the set of action labels, containing the special label τ accounting for silent actions; $\rightarrow$ is the transition relation. As usual, we write $s_1 \xrightarrow{\mu} s_2$ when $(s_1, \mu, s_2) \in \rightarrow$. The definition of weak bisimilarity $\approx$ is as in Section 2. We omit explicit reference to $\mathcal{T}$ when there is no ambiguity.

We reason about *state functions*, i.e., functions from S to S, and use f, f', g to range over them. We recall that $(f \circ g)(x) = f(g(x))$ for all x. The CCS processes of Section 2 correspond here to the states of an LTS; and CCS contexts correspond to state functions.

▶ **Definition 12** (Autonomy)**.** For state functions f, f' we say that there is an *autonomous μ-transition from f to f'*, written $f \xrightarrow{\mu} f'$, if for all states x it holds that $f(x) \xrightarrow{\mu} f'(x)$.

Likewise, given a set $\mathcal{F}$ of state functions and $f \in \mathcal{F}$, we say that a transition $f(x) \xrightarrow{\mu} y$ is *autonomous on $\mathcal{F}$* if, for some $f' \in \mathcal{F}$ we have $f \xrightarrow{\mu} f'$ and $y = f'(x)$. Moreover, we say that *function f is autonomous on $\mathcal{F}$* if all the transitions emanating from f (that is, all transitions of the form $f(x) \xrightarrow{\mu} y$, for some x, μ, y) are autonomous on $\mathcal{F}$.

When $\mathcal{F}$ is clear, we omit it, and we simply say that a function *is autonomous*.

Thus, f is autonomous on $\mathcal{F}$ if, for some indexing set I, there are μ_i and $f_i \in \mathcal{F}$ such that for all x we have $f(x) \xrightarrow{\mu_i} f_i(x)$, for each i; the set of all transitions emanating from $f(x)$ is precisely $\cup_i \{f(x) \xrightarrow{\mu_i} f_i(x)\}$. Autonomous functions correspond to CCS guarded contexts, which do not need their process argument to perform the first transition. We now formulate conditions under which, intuitively, a state function behaves like a CCS context. Functions satisfying these conditions are called *operators*.

▶ **Definition 13** (Set of operators)**.** Consider an LTS $\mathcal{T}$, and a set $\mathcal{O}$ of functions from S to S. We say that $\mathcal{O}$ is a *set of operators on $\mathcal{T}$* if the following conditions hold:

1. $\mathcal{O}$ contains the identity function;
2. $\mathcal{O}$ is closed by composition (that is, $f \circ g \in \mathcal{O}$ whenever $f, g \in \mathcal{O}$);
3. composition preserves autonomy (i.e., if g is autonomous on $\mathcal{O}$, then so is $f \circ g$);
4. all functions in $\mathcal{O}$ respect $\approx$, i.e., $x \approx y$ and $f \in \mathcal{O}$ imply $f(x) \approx f(y)$.

A 'symmetric variant' of clause 3 always holds: if f is autonomous, then so is $f \circ g$. The autonomous transitions of a set of operators yield an LTS whose states are the operators themselves. Such transitions are of the form $f \xrightarrow{\mu} g$. Where the underlying set $\mathcal{O}$ of operators is clear, we simply call a function belonging to $\mathcal{O}$ an *operator*.

We use state functions to express equations, such as $X = f(X)$. We thus look at conditions under which such an equation has a unique solution (again, the generalisation to a system of equations is easy).

Thinking of functions as equation expressions, to formulate our abstract theory about unique solution of equations, we have to define the divergences of finite and infinite unfoldings of state functions. The nth unfolding of f (for $n \geq 1$), f^n, is the function obtained by n applications of f. An operator is *well-behaved* if there is n with f^n autonomous (the well-behaved operators correspond, in CCS, to equations some finite unfolding of which yields a guarded expression). We also have to reason about the infinite unfolding of an equation $X = f(X)$. For this, given a set $\mathcal{O}$ of operators, we consider the infinite terms obtained by infinite compositions of operators in $\mathcal{O}$, that is, the set coinductively defined by the grammar:

$$F := f \circ F \qquad \text{where } f \in \mathcal{O} \text{ (i.e., } f \text{ is a metavariable for the elements in } \mathcal{O}).$$

(We do not need finite compositions, as $\mathcal{O}$ itself is closed under finite compositions.) In particular, we write f^∞ for the infinite term $f \circ f \circ f \circ \ldots$.

The autonomous transitions for such infinite terms are given by the following rules:

$$\frac{g \xrightarrow{\mu} g'}{g \circ F \xrightarrow{\mu} g' \circ F}\ g \text{ autonomous} \qquad \frac{(g \circ f) \circ F \xrightarrow{\mu} F'}{g \circ (f \circ F) \xrightarrow{\mu} F'}\ g \text{ not autonomous}$$

Intuitively a term is 'unfolded', with the second rule, until an autonomous function is uncovered, and then transitions are computed using the first rule (we disallow unnecessary unfoldings; these would complicate our abstract theorems, by adding duplicate transitions, since the transitions of $g \circ f$ duplicate those of g when g is autonomous). An infinite term has no transitions if none of its finite unfoldings ever yields an autonomous function. This situation does not arise for terms of the form f^∞ or $g \circ f^\infty$, where f is well-behaved, which are the terms we are interested in. Note that no infinite term belongs to a set of operators.

▶ **Definition 14** (Operators and divergences)**.** Let f, f', f_i be operators in a set $\mathcal{O}$ of operators, and consider the LTS induced by the autonomous transitions of operators in $\mathcal{O}$. A sequence of transitions $f_1 \xrightarrow{\mu_1} f_2 \xrightarrow{\mu_2} f_3 \ldots$ is a *divergence* if for some $n \geq 1$ we have $\mu_i = \tau$ whenever $i \geq n$. We also say that f_1 *diverges*. We apply these notations and terminology also to infinite terms (such as $f_1 \circ f_2 \circ \ldots$), as expected.

In the remainder of the section we fix a set $\mathcal{O}$ of operators and we only consider autonomous transitions on $\mathcal{O}$. We now state the "abstract version" of Theorem 6 (the proof being similar).

▶ **Theorem 15** (Unique solution, abstract formulation)**.** *Let f be a well-behaved operator on $\mathcal{O}$. If f^∞ does not diverge, then the equation $X = f(X)$ either has no solution or has a unique solution for $\approx$.*

The equation in the statement of the theorem might have no solution at all. For an example, consider the LTS $(\mathbb{N}, \{a\}, \rightarrow)$ where for each n we have $n+1 \xrightarrow{a} n$. The function f with $f(n) = n+1$ is an operator of the set $\mathcal{O} = \{f^n\}_{n \in \mathbb{N}}$ (with $f^0 = \mathtt{Id}$, the identity function). Function f is autonomous because, for all n, the only transition of $f(n)$ is $f(n) \xrightarrow{a} n$ (this transition is autonomous because $f \xrightarrow{a} \mathtt{Id}$). A fixpoint of f would be an element x with $x \xrightarrow{a} x$, and there is no such x in the LTS.

Theorem 15 can be refined along the lines of Theorem 8. For this, we have to relate the divergences of any f^n (for $n \geq 1$) to divergences of f^∞, in order to distinguish between innocuous and non-innocuous divergences. To build a divergence of f^∞ from a divergence of f^n, for $n \geq 1$, we need to reason up to (finite) unfoldings of f: thus we set $=_f$ to be the symmetric reflexive transitive closure of the relation that relates g and g' whenever $g = g' \circ f$.

▶ **Lemma 16.** *Consider an autonomous operator f on $\mathcal{O}$ and a divergence of f^n*

$$f^n \xrightarrow{\mu_1} f_1 \xrightarrow{\mu_2} \ldots \xrightarrow{\mu_i} f_i \xrightarrow{\tau} f_{i+1} \xrightarrow{\tau} \ldots$$

This yields a divergence of f^∞*:* $f^\infty \xrightarrow{\mu_1} g_1 \circ f^\infty \xrightarrow{\mu_2} \dots \xrightarrow{\mu_i} g_i \circ f^\infty \xrightarrow{\tau} g_{i+1} \circ f^\infty \xrightarrow{\tau} \dots$ *such that for all* $i \geq 1$*,* g_i *is an operator and* $g_i =_f f_i$*.*

Given a divergence Δ of f^n, we write Δ^∞ to indicate the divergence of f^∞ obtained from Δ as in Lemma 16. We call *innocuous* a divergence of f^∞ that can be described in this way, that is, a divergence Φ such that $\Phi = \Delta^\infty$ for some n and some divergence Δ of f^n.

▶ **Theorem 17** (Unique solution with innocuous divergences, abstract formulation)**.** *Let* $f \in \mathcal{O}$ *be a well-behaved operator. If all divergences of* f^∞ *are innocuous, then the equation* $X = f(X)$ *either has no solution or has a unique solution for* $\approx$*.*

4.2 Reasoning with other Behavioural Relations

Trace-based Equivalences. We can adapt the results of the previous section about bisimilarity to other settings, including both preorders and non-coinductive relations. As an example, we consider trace-based relations. We write $\approx_{tr}$ for trace equivalence, that equates two processes having the same set of (finite) traces. The definitions from previous sections are the same as for $\approx$, replacing $\approx$ with $\approx_{tr}$. All theorems presented for $\approx$ can be adapted to $\approx_{tr}$, and the proofs are similar. As an example, Theorem 17 becomes:

▶ **Theorem 18.** *Let* $f \in \mathcal{O}$ *be a well-behaved operator. If all divergences of* f^∞ *are innocuous, then the equation* $X = f(X)$ *either has no solution or has a unique solution for* $\approx_{tr}$*.*

In contrast, the theorems fail for *infinitary trace equivalence*, $\approx_{tr}^\infty$ (whereby two processes are equated if they have the same traces, including the infinite ones), for the same reason why the 'unique solution of contraction' technique fails in this case [25]. As a counterexample, we consider equation $X = a + a.X$, whose syntactic solution has no divergences. The process $P \stackrel{\triangle}{=} \sum_{n \in \mathbb{N}^*} a^n$ is a solution, yet it is not $\approx_{tr}^\infty$-equivalent to the syntactic solution of the equation, that is capable of performing infinitely many a-transitions.

Preorders. We show how the theory for equivalences can be transported onto *preorders*. This means moving to *systems of preequations*, $\{X_i \leq E_i\}_{i \in I}$. With preorders, our theorems have a different shape: we do not use preequations to reason about unique solution – we expect interesting preequations to have many solutions, some of which may be incomparable with each other. We rather derive theorems to prove that, in a given preorder, any solution of a preequation is below its syntactic solution.

▶ Remark. The opposite direction for preequations, namely $\{X_i \geq E_i\}_{i \in I}$ is less interesting. It would mean aiming to prove that the syntactic solution is below other solutions. This is usually a trivial property for a behavioural preorder, without any hypothesis such as autonomy or non divergence (a possible exception is a preorder with infinitary observables, such as infinitary trace inclusion, where the property may fail).

We write $\subseteq_{tr}$ for trace inclusion, $\subseteq_{tr^\infty}$ for infinitary trace inclusion, and $\leq_s$ for weak simulation. These preorders are standard from the literature [28].

In the abstract setting, the body of the preequations are functions. Then the theorems give us conditions under which, given a preequation $X \leq f(X)$ and a behavioural preorder $\preceq$, a solution r, i.e., a state for which $r \preceq f(r)$ holds, is below the syntactic solution f^∞. We present the counterpart of Theorem 17; other theorems are transported in a similar manner.

▶ **Theorem 19.** *Let* $f \in \mathcal{O}$ *be a well-behaved operator. If all divergences of* f^∞ *are innocuous, then, given* $\leq \in \{\subseteq_{tr}, \subseteq_{tr^\infty}, \leq_s\}$*, whenever* $x \leq f(x)$ *we also have* $x \leq f^\infty$*, for any state* x*.*

Theorem 19 intuitively says that the syntactic solution of a preequation is *maximal* among all solutions. Note that, in contrast with equations, Theorem 19 and the theory of preequations also work for *infinitary trace inclusion*.

4.3 Rule formats

A way to instantiate the results in Sections 4.1 and 4.2 is to consider *rule formats* [1, 16]. These provide a specification for the form of the SOS rules used to describe the constructs of a language. To fit a rule format into the abstract formulation of the theory from Section 4.1, we view the constructs of a language as functions on the states of the LTS (the processes of the language). One of the most common formats is GSOS [3, 29, 9]. For lack of space we only show the instantiation to GSOS of Theorem 8. In the statement, we consider an extension of a GSOS language with constants, in the same way as they appear in CCS, so that the definitions of 'syntactic solution of equations' and of 'innocuous divergence' can be taken to be the same as for CCS (these are easier to grasp than their formulation in Section 4.1).

▶ **Theorem 20.** *Consider a language whose constructs have SOS rules in the GSOS format and preserve $\approx$, and an equation $X = E$ for the language. If E is autonomous (over the set of functions corresponding to the contexts of the language), and if, in the language extended with constants, the syntactic solution of the equation only has innocuous divergences, then either the equation has no solution or it has a unique solution for $\approx$.*

We briefly discuss the hypotheses in the theorem. Some GSOS rule formats guarantee congruence for weak bisimilarity [29, 9], which allows one to remove the corresponding condition (the condition could actually we weakened, by considering the syntactic positions in which the variables can occur in the equations). Checking the autonomy property is often straightforward; for instance, it holds if, in the body of an equation, all variables are underneath an axiom construct, that is, a construct that (like prefix in CCS) is defined by means of SOS rules in which the set of premises is empty. In the tyft/tyxt formats [11], lookaheads are possible. Lookahead allows one to write rules that 'look into the future' (a transition is allowed if certain sequences of actions are possible); this breaks autonomy (condition 3 of Definition 13), hence Theorem 17 is not applicable; see Appendix C.

5 Name Passing: the π-calculus

5.1 Unique solution in the asynchronous π-calculus

In this section, we port our results onto the asynchronous π-calculus, $A\pi$ (addressing the full π-calculus would also be possible, but somewhat more involved). To allow constants (recursive definitions) and equations, we enrich the syntax of $A\pi$ [26] with parametrised processes $(\widetilde{a})\,P$. These and the constants form the set of *abstractions*, ranged over by F, G. We omit the definitions of free and bound names. An expression is *closed* if it does not have free names. Constant definitions are of the form $K \stackrel{\triangle}{=} F$, where F is a closed abstraction. The grammar for processes includes also the application construct $F\langle\widetilde{a}\rangle$, used to instantiate the formal parameters of the abstraction F with the actual parameters $\widetilde{a}$.

The body of an equation is also a closed abstraction, possibly containing equation variables. Since the calculus is polyadic, we rely on a sorting system [15] to avoid disagreements in the arities of the tuples of names carried by a given name, and in the parameters of abstractions and equations. For lack of space, we omit the full grammar and the operational semantics (see

Appendix D). When writing examples, for readability, we assume that the syntax contains the guarded replication operator $!a(\widetilde{b}).\, P$ (it could be encoded, using constants).

In bisimulations or similar coinductive relations for the asynchronous π-calculus, no name instantiation is required in the input clause or elsewhere (provided α-convertible processes are identified); i.e., the *ground* versions of the relations are congruence relations [26]. Similarly, the extension of bisimilarity to abstractions only considers fresh names: $F \approx F'$ if $F\langle\widetilde{a}\rangle \approx F'\langle\widetilde{a}\rangle$ where $\widetilde{a}$ is a tuple of fresh names (as usual, of the appropriate sort).

Theorems 6 and 8 for CCS can be adapted to the asynchronous π-calculus. The definitions concerning transitions and divergences are transported to $A\pi$ as expected. In the case of an abstraction, one first has to instantiate the parameters with fresh names; thus F has a divergence if the process $F\langle\widetilde{a}\rangle$ has a divergence, where $\widetilde{a}$ are fresh names.

▶ **Theorem 21** (Unique solution in Aπ)**.** *A guarded system of equations whose syntactic solutions do not contain divergences has a unique solution for* $\approx$*.*

▶ **Theorem 22** (Unique solution with innocuous divergences in Aπ)**.** *A guarded system of equations whose syntactic solutions only have innocuous divergences has a unique solution for* $\approx$*.*

We pointed out in earlier sections on CCS the connection between techniques based on unique solution of equations and 'up to context' enhancements of the bisimulation proof method. The same connection is less immediate in name-passing calculi, where indeed there are noticeable differences. In particular, 'up to context' enhancements for the ground bisimilarity of the π-calculus require closure under name instantiation, even when ground bisimilarity is known to be preserved by substitutions (it is an open problem whether the closure can be lifted). Thus, when comparing two derivatives $C[P]$ and $C[Q]$, in general it is not sufficient that P and Q alone are in the candidate relation: one is required to include also all their closures under name substitutions (or, if the terms in the holes are abstractions, instantiation of their parameters with arbitrary tuples of names). In contrast, the two unique solution theorems above are 'purely ground': $F = (\widetilde{x})P$ is solution of an equation $X = (\widetilde{x})E$ if P and $E\{^F/_X\}$ are ground bisimilar – a single ground instance of the equation is evaluated.

5.2 An application: encoding of the call-by-name λ-calculus

As an extended example of application of the techniques proposed in the π-calculus, we revisit the proof of the full abstraction of Milner's encoding of the call-by-name (or lazy) λ-calculus into $A\pi$ [14] with respect to Lévy Longo Trees (LTs), precisely the completeness part. We use M, N to range over the set Λ of λ-terms, and x, y, z to range over λ variables. If M is an open λ-term, then either M diverges, or $M \Rightarrow \lambda x.\, M'$, or $M \Rightarrow x\ M_1\ \ldots\ M_n$.

▶ **Definition 23** (Lévy-Longo Tree)**.** The *Lévy-Longo Tree* (LT) of an open λ-term M, written $LT(M)$, is the (possibly infinite) tree defined coinductively as follows.

1. If M diverges, then $LT(M)$ is the tree with a single node labelled $\bot$.
2. If $M \to \lambda x.\, M'$, then $LT(M)$ is the tree with a root labelled with "$\lambda x.$", and $LT(M')$ as its unique descendant.
3. If $M \to x\ M_1\ \ldots\ M_n$, then $LT(M)$ is the tree with a root labelled with "x", and $LT(M_1), \ldots, LT(M_n)$ (in this order) as its n descendants.

LT equality (two λ-terms are identified if their LTs are equal) can also be presented as a bisimilarity (*open bisimilarity*, $\simeq^o$), defined as the largest *open bisimulation*.

▶ **Definition 24.** A relation $\mathcal{R}$ on Λ is an *open bisimulation* if, whenever $M\ \mathcal{R}\ N$:

1. $M \Rightarrow \lambda x.\, M'$ implies $N \Rightarrow \lambda x.\, N'$ with $M' \mathcal{R} N'$;
2. $M \Rightarrow x\ M_1\ \ldots\ M_n$ with $n \geq 0$ implies $N \Rightarrow x\ N_1\ \ldots\ N_n$ and $M_i \mathcal{R} N_i$ for all $1 \leq i \leq n$.
3. The converse of clauses 1 and 2 on the challenges from N.

Milner's encoding is defined thus: $[\![\lambda x.\, M]\!] \triangleq (p)\ p(x,q).\ [\![M]\!]\langle q\rangle$ $\quad [\![x]\!] \triangleq (p)\ \overline{x}\langle p\rangle$
$[\![M\ N]\!] \triangleq (p)\ \boldsymbol{\nu} r, x\ ([\![M]\!]\langle r\rangle \mid \overline{r}\langle x, p\rangle \mid\ !x(q).\, [\![N]\!]\langle q\rangle)$

The full abstraction theorem for the encoding [24, 27] states that two λ-terms have the same LT iff their encodings into $A\pi$ are weakly bisimilar terms. Full abstraction has two components: soundness, which says that if the encodings are weakly bisimilar then the original terms have the same LT; and completeness, which is the converse direction. The proof [24] first establishes some operational correspondence between the behaviour (visible and silent actions) of λ-terms and of their encodings. Then, exploiting this correspondence, soundness and completeness are proved using the bisimulation proof method. For soundness, this is just open bisimulation (Definition 24). In contrast, completeness involves enhancements of the proof method, notably 'bisimulation up to context and expansion'. As a consequence, the technique requires having developed the basic theory for the expansion preorder (e.g., precongruence properties and basic algebraic laws), and requires an operational correspondence fine enough in order to be able to reason about expansion.

Below we show that, by appealing to unique solution of equations, completeness can be proved by defining an appropriate system of equations, each of which having a simple shape, and without the need for auxiliary preorders. For this, the only results needed are: (i) validity of β-reduction for the encoding (Lemma 25), whose proof is simple and consists in the application of a few algebraic laws (including laws for replication); (ii) the property that if M diverges then $[\![M]\!]\langle p\rangle$ may never produce a visible action [24].

▶ **Lemma 25** (Validity of β-reduction, [24]). *For $M \in \Lambda$, if $M \to M'$ then $[\![M]\!] \approx [\![M']\!]$.*

▶ **Theorem 26** (Full abstraction, [24]). *For $M, N \in \Lambda$, $[\![M]\!] \approx [\![N]\!]$ iff $LT(M) = LT(N)$.*

Suppose V and W are two λ-terms with the same LT. We define a system of equations, solutions of which are obtained from the encodings of V and W. We will then use Theorem 22 to deduce $[\![V]\!] \approx [\![W]\!]$. If V and W have the same LT, then there is an open bisimulation $\mathcal{R}$ containing the pair (V, W). The variables of the equations are of the form $X_{M,N}$ for $M \mathcal{R} N$, and there is one equation for each pair in $\mathcal{R}$. We show how the equation for a pair $M \mathcal{R} N$ is built. We assume an ordering of the λ-calculus variables so to be able to view a finite set of variables as a tuple. Thus we write $\widetilde{x}$ for the variables appearing free in M or N.

Essentially, the equations are the translation of the clauses of Definition 24, assuming a generalisation of the encoding to equation variables:

If M, N are both divergent, then the equation is $X_{M,N} = (\widetilde{x}, p)!\tau$.

If M, N satisfy clause 1 of Definition 24, the equation is $X_{M,N} = (\widetilde{x}, p)p(x,q).\, X_{M',N'}\langle \widetilde{y}, q\rangle$, where $\widetilde{y}$ are the free variables in M', N'.

Finally, if M, N satisfy clause 2 of Definition 24, the equation is given by the translation of $x\ X_{M_1,N_1} \ldots X_{M_n,N_n}$, which, rearranging restrictions and parallel compositions, is

$$X_{M,N} = (\widetilde{x}, p)(\boldsymbol{\nu} r_0, \ldots, r_n\,)(\ \overline{x}\langle r_0\rangle \mid \overline{r_0}\langle r_1, x_1\rangle \mid \ldots \mid \overline{r_{n-1}}\langle r_n, x_n\rangle \mid\ !x_1(q_1).\, X_{M_1,N_1}\langle \widetilde{x_1}, q_1\rangle \mid \ldots \mid\ !x_n(q_n).\, X_{M_n,N_n}\langle \widetilde{x_n}, q_n\rangle)\ .$$

where $\widetilde{x_i}$ are the free variables in M_i, N_i. Applying Lemma 9 (more precisely, its $A\pi$ analogue, with additional reference to the sorting system to handle name instantiation in abstractions and inputs), we show that the equations may only produce innocuous divergences.

Now, for $(M, N) \in \mathcal{R}$, set $F_{M,N}$ to be the abstraction $(\widetilde{x}, p)[\![M]\!]\langle p\rangle$, and similarly $G_{M,N} \triangleq (\widetilde{x}, p)[\![N]\!]\langle p\rangle$. The set of all such abstractions $F_{M,N}$ yields a solution for the system of equations, and the same for the $G_{M,N}$'s. The proof that they are solutions (hence bisimilar) is a consequence of Lemma 25. For instance, for clause 1 of Definition 24, we have:

$$\begin{array}{rcl} F_{M,N} & = & (\widetilde{x}, p)[\![M]\!]\langle p\rangle \\ & \approx & (\widetilde{x}, p)[\![\lambda x.\, M']\!]\langle p\rangle \;\approx\; (\widetilde{x}, p)[\![\lambda x.\, X_{M',N'}]\!]\langle p\rangle\{^{F_{M',N'}}/_{X_{M',N'}}\} \end{array}$$

6 Future Work

We have compared our techniques to one of the most powerful forms of enhancements of the bisimulation proof method, namely Pous 'up to transitivity and context', showing that, up to a technical condition, our techniques are at least as powerful. We believe that also the converse holds, though possibly under different side conditions. We leave a detailed analysis of this comparison, which seems non-trivial, for future work. In this respect, the goal of the work on unique solution of equations is to provide a way of better understanding up-to techniques and to shed light into the conditions for their soundness. Pous technique, for instance, is arguably more complex, both in its definition and its application.

Another aspect in which a deep comparison with up-to techniques might be useful is understanding the need for the closure under substitutions in up to context techniques for name-passing calculi such as the asynchronous π-calculus. Such a closure is rather heavy and is a long-standing open problem. However, currently it is unclear how to formally relate bisimulation enhancements and 'unique solution of equations' in name-passing calculi.

It would also be interesting to study whether the connections between up-to techniques and unique-solution techniques can be phrased using the abstract formulation of Section 4. Up-to techniques have been analysed in an abstract setting using lattice theory [19] and category theory [4, 23].

In comparison with the enhancements of the bisimulation proof method, the main drawback of the techniques exposed in this paper and based on unique solution of equations is the presence of a semantic condition, involving divergence: the unfoldings of the equations should not produce divergences, or only produce innocuous divergences. A syntactic condition for this has been proposed (Lemma 9). Various techniques for checking divergence exist in the literature, including type-based techniques [30, 7]. However, in general divergence is undecidable, and therefore, the check may sometimes be unfeasible. Nevertheless, the equations that one writes for proofs usually involve forms of 'normalised' processes, and as such they are divergence free (or at most, contain only innocuous divergences). More experiments are needed to validate this claim or to understand how limiting this problem is.

References

1 L. Aceto, W.J. Fokkink, and C. Verhoef. *Handbook of Process Algebra (J.A. Bergstra and A. Ponse and S.A. Smolka, editors)*, chapter Structural operational semantics. Elsevier Science, 2001.

2 J.C.M. Baeten, T. Basten, and M.A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*. Cambridge University Press, 2010.

3 Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages*, pages 229–239, 1988.

4 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. A general account of coinduction up-to. *Acta Inf.*, 54(2):127–190, 2017.

5 Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

6 Stephen D. Brookes and A. W. Roscoe. An improved failures model for communicating processes. In *Seminar on Concurrency, Carnegie-Mellon University, Pittsburg, PA, USA*, pages 281–305, 1984.

7 Romain Demangeon, Daniel Hirschkoff, and Davide Sangiorgi. Termination in impure concurrent languages. In *CONCUR 2010 - Concurrency Theory, 21th International Conference*, pages 328–342, 2010.

8 A. Durier. Divergence and unique solution of equations in an abstract setting, Coq formal proof. Available at `http://perso.ens-lyon.fr/adrien.durier/uniquesolution/`, 2017.

9 Wan Fokkink and Rob J. van Glabbeek. Divide and congruence II: delay and weak bisimilarity. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, pages 778–787. ACM, 2016.

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

11 Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inf. Comput.*, 100(2):202–260, 1992.

12 C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

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

14 Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

15 Robin Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999.

16 Mohammad Reza Mousavi, Michel A. Reniers, and Jan Friso Groote. SOS formats and meta-theory: 20 years after. *Theor. Comput. Sci.*, 373(3):238–272, 2007.

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

18 Damien Pous. *Up to techniques for bisimulations*. PhD thesis, ENS Lyon, February 2008.

19 Damien Pous. Coinduction all the way up. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, pages 307–316, 2016.

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

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

22 A. W. Roscoe. *Understanding Concurrent Systems*. Springer, 2010.

23 Jurriaan Rot, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Coalgebraic bisimulation-up-to. In *SOFSEM 2013: Theory and Practice of Computer Science, 39th International Conference on Current Trends in Theory and Practice of Computer Science*, pages 369–381, 2013.

24 D. Sangiorgi. Lazy functions and mobile processes. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000.

25 Davide Sangiorgi. Equations, contractions, and unique solutions. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015*, pages 421–432, 2015.

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

27 Davide Sangiorgi and Xian Xu. Trees from functions as processes. In *25th International Conference, CONCUR 2014, Rome, Italy*, LNCS, pages 78–92. Springer Verlag, 2014.

28 Rob J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In *CONCUR '90, Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, August 27-30, 1990, Proceedings*, pages 278–297, 1990.

29 Rob J. van Glabbeek. On cool congruence formats for weak bisimulations. In *Theoretical Aspects of Computing - ICTAC 2005, Second International Colloquium*, pages 318–333, 2005.

30 N. Yoshida, M. Berger, and K. Honda. Strong Normalisation in the Pi-Calculus. *Information and Computation*, 191(2):145–202, 2004.

A Proofs in CCS

The goal of this appendix is to give intuitions about the schema used for the proofs of Theorems 6 and 8. Only sketches are presented, that do not contain all the details.

Main theorem. As equation expressions are terms of an extended CCS grammar, that includes variables, we can apply the SOS rules of CCS to them (assuming that variables have no transitions). We extend accordingly to expressions notations and terminology for LTS and transitions.

We have the following properties for transitions of processes of the form $E[P]$, where the transition emanates from the E component only:

▶ **Lemma 27** (Expression transitions)**.**

1. *Given E and E' two equation expressions, if $E \xrightarrow{\mu} E'$, then $E[P] \xrightarrow{\mu} E'[P]$, for all process P, and $E[E''] \xrightarrow{\mu} E'[E'']$ for all equation expression E''.*
2. *If E is a guarded expression and $E[P] \xrightarrow{\mu} T$, then there is an expression E' such that $E \xrightarrow{\mu} E'$ and $T = E'[P]$. Similarly for a transition $E[E'] \xrightarrow{\mu} E''$.*

In the hypothesis of case 1 above, we sometimes call a transition $E[P] \xrightarrow{\mu} E'[P]$ *an instance* of the expression transition $E \xrightarrow{\mu} E'$.

▶ **Definition 28** (Reducts)**.** The *set of reducts* of an expression E, written $\mathtt{red}(E)$, is given by:

$$\mathtt{red}(E) \quad \triangleq \quad \bigcup_n \{E_n \mid E \xrightarrow{\mu_1} E_1 \cdots \xrightarrow{\mu_n} E_n \text{ for some } \mu_i, E_i \ (1 \leq i \leq n)\}.$$

The set of *reducts of the unfoldings* of E is defined as $\mathtt{red}_\omega(E) \triangleq \cup_{n\in\mathbb{N}} \mathtt{red}(E^n)$.

▶ **Definition 29.** An equation expression E *protects its solutions* if, for all solution P of the equation $X = E$, the following holds: for all $E' \in \mathtt{red}_\omega(E)$, if $E'[P] \stackrel{\mu}{\Rightarrow} P'$ for some μ and P', then there exists E'' and n such that $E'[E^n] \stackrel{\hat{\mu}}{\Rightarrow} E''$, and $E''[P] \approx P'$.

In other words, E protects its solutions when all sequences of transitions emanating from $E'[P]$, where P is a solution of E and E' is a reduct of the unfoldings of E, can be mimicked by transitions involving unfoldings of E and reducts of E only (without P performing a transition). This condition ensures unique solution.

▶ **Proposition 30.** *If the equation expression E protects its solutions, then the equation $X = E$ has a unique solution for $\approx$.*

Proof. Given two solutions P, Q of the equation $X = E$, we prove that the relation

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

is a bisimulation relating P and Q.

We consider $(S,T) \in \mathcal{R}$, that is, $S \approx E'[P]$ and $T \approx E'[Q]$, for some $E' \in \mathtt{red}_\omega(E)$. If $S \xrightarrow{\mu} S'$, then by bisimilarity $E'[P] \stackrel{\hat{\mu}}{\Rightarrow} S'' \approx S'$. Since E protects its solutions, there are n, E'' such that $E'[E^n] \stackrel{\hat{\mu}}{\Rightarrow} E''$ and $E''[P] \approx S''$. We can deduce by Lemma 27(1) that $E'[E^n[Q]] \stackrel{\hat{\mu}}{\Rightarrow} E''[Q]$. Since Q is a solution of E, we have $Q \approx E^n[Q]$. This entails, as we know by hypothesis $T \approx E'[Q]$, that $T \approx E'[Q] \approx E'[E^n[Q]]$. By bisimilarity, we deduce from $E'[E^n[Q]] \stackrel{\hat{\mu}}{\Rightarrow} E''[Q]$ that $T \stackrel{\hat{\mu}}{\Rightarrow} T' \approx E''[Q]$.

The situation can be depicted on the following diagram:

$$\begin{array}{ccccccccccc}
S & \approx & E'[P] & \approx & E'[E^n[P]] & & E'[E^n[Q]] & \approx & E'[Q] & \approx & T \\
{\scriptstyle\mu}\downarrow & & {\scriptstyle\hat{\mu}}\Downarrow & & {\scriptstyle\hat{\mu}}\Downarrow & = & {\scriptstyle\hat{\mu}}\Downarrow & & & & {\scriptstyle\hat{\mu}}\Downarrow \\
S' & \approx & S'' & \approx & E''[P] & & E''[Q] & & \approx & & T'
\end{array}$$

Then, from $E' \in \mathtt{red}_\omega(E)$ and $E'[E^n] \stackrel{\hat{\mu}}{\Longrightarrow} E''$, we deduce that $E'' \in \mathtt{red}_\omega(E)$. Finally, $T' \approx E''[Q]$ and $S' \approx E''[P]$ give that $(S,T) \in \mathcal{R}$.

We reason symmetrically for $T \xrightarrow{\mu} T'$. □

The previous definitions and statements were given for a single equation for simplicity of presentation, but they can be extended to systems of equations. Given a system of equations $\widetilde{E}$, we write $\widetilde{E}^2$ for the system $\{X_i = E_i[\widetilde{E}]\}_{i\in I}$, and similarly for $\widetilde{E}^n$. We define $\widetilde{K}_{\widetilde{E}}$ to be the set of constants $\{K_{\widetilde{E},i} \triangleq E_i[\widetilde{K}_{\widetilde{E}}]\}_{i\in I}$, and we call these the syntactic solutions of $\widetilde{E}$.

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

We recall the statement of Theorem 6 for systems of equations:

▶ Theorem (Theorem 6, Section 3). A guarded system of equations whose syntactic solutions do not diverge has a unique solution for $\approx$.

Proof (Sketch, case of a single equation). Given a guarded equation expression E, we prove that E protects the solutions of its equations.

To prove that, we need to consider a transition $E_0[P] \stackrel{\mu}{\Rightarrow} P'$, for some $E_0 \in \mathtt{red}_\omega(E)$ and some solution P of E.

We build a sequence of expressions E_n and an increasing sequence of transitions $E_0[E^n] \stackrel{\mu?}{\Longrightarrow} E_n$ (where $\stackrel{\mu?}{\Longrightarrow} \stackrel{\text{def}}{=} \Rightarrow \cup \stackrel{\hat{\mu}}{\Rightarrow}$) such that: either this construction stops, yielding a transition $E_0[E^n] \stackrel{\mu}{\Longrightarrow} E_n$, or the construction is infinite, therefore giving a divergence of K_E.

We build this sequence so that it additionally satisfies:

- Either we have $E_0[E^n] \stackrel{\hat{\mu}}{\Rightarrow} E_n$ and $E_n[P] \Rightarrow\approx P'$, or $E_0[E^n] \Rightarrow E_n$ and $E_n[P] \stackrel{\hat{\mu}}{\Rightarrow}\approx P'$.
- The sequence is strictly increasing: the sequence of transitions $E_0[E^{n+1}] \stackrel{\mu?}{\Longrightarrow} E_n[E]$ is a strict prefix of the sequence of transitions $E_0[E^{n+1}] \stackrel{\mu?}{\Longrightarrow} E_{n+1}$

We initialise with the empty sequence from E_0.

Then, at step n, suppose we have, for example, $E_0[E^n[P]] \stackrel{\mu}{\Rightarrow} E_n[P] \Rightarrow T_n$, with $P' \approx T_n$.

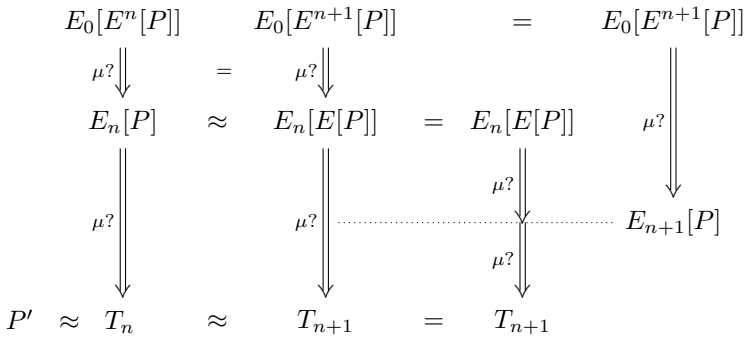

Figure 2 Recursion: construction of the sequence of transitions $E_0[E^n[P]] \stackrel{\mu?}{\Longrightarrow} E_n$

- If $E_n[P] \Rightarrow T_n$ is the empty sequence, we stop. We have in this case $E_0[E]^n \stackrel{\mu}{\Rightarrow} E_n$ and $P' \approx E_n[P]$.
- Otherwise (as depicted on Figure 2), we unfold equation E once more: we have $E_0[E^{n+1}] \stackrel{\mu}{\Rightarrow} E_n[E]$ by Lemma 27. By congruence and bisimilarity we have $E_n[E[P]] \Rightarrow T_{n+1} \approx P'$ for some T_{n+1}. If $E_n[E[P]] \Rightarrow T_{n+1}$ is the empty sequence, we stop as previously. Otherwise, we take $E_n[E] \stackrel{\mu?}{\Longrightarrow} E_{n+1}$ to be the longest prefix sequence of transitions in $E_n[E[P]] \Rightarrow T_{n+1}$ that are instances of expression transitions from $E_n[E]$ (we remark that as $E_n[E]$ is guarded, this sequence is not empty).

Suppose now that the construction given above never stops. We know that $E_n[E] \stackrel{\mu?}{\Longrightarrow} E_{n+1}$, therefore $E_n[K_E] \stackrel{\mu?}{\Longrightarrow} E_{n+1}[K_E]$. This gives an infinite sequence of transitions starting from $E_0[K_E]$: $E_0[K_E] \stackrel{\mu?}{\Longrightarrow} E_1[K_E] \stackrel{\mu?}{\Longrightarrow} \ldots$. We observe that in the latter sequence, every step involves at least one transition, and moreover, there is at most one visible action (μ) occurring in this infinite sequence. Therefore $E_0[K_E]$ is divergent, which contradicts the hypothesis of the theorem. Hence, the construction does stop, and this concludes the proof. □

Innocuous divergences. We recall the statement of Theorem 8:

▶ Theorem (Theorem 8, Section 3). Let $\widetilde{X} = \widetilde{E}$ be a system of guarded equations, and $\widetilde{K}_{\widetilde{E}}$ be its syntactic solutions. If all divergences of any $K_{\widetilde{E},i}$ are innocuous, then $\widetilde{E}$ has a unique solution for $\approx$.

Proof (Sketch, case of a single equation). We reason like in the proof of Theorem 6. Along the construction in that proof, if at some point the transition $E_0[E^n[P]] \stackrel{\hat{\mu}}{\Rightarrow} T_n$ is an instance of an expression transition $E_0[E^n] \stackrel{\hat{\mu}}{\Rightarrow} E_n$, then the construction can stop.

Consequently, if the construction never stops, we build a non innocuous divergence: we can assume that for any n, $E_0[E^n[P]] \stackrel{\hat{\mu}}{\Rightarrow} T_n$ is not an instance of an expression transition from $E_0[E^n]$. This means that in this sequence of transitions, the LTS rule `const` is used at least n times applied to the constant K_E. Hence, the divergence we build uses at least n times the rule `const` applied to the constant K_E, for all n. Therefore, the divergence is not innocuous, which is a contradiction. □

B Completeness with respect to up-to techniques

As above, the goal of this appendix is to give some intuitions about Theorem 11 and its proof, and not to provide all the details of said proof.

Formally, contexts are processes that can have holes, numbered by some finite set of integers. Therefore, they are like equation expressions, except that numbered holes are used instead of variable names. In this section, we will switch freely between equation expressions and contexts, as contexts are needed to study up-to techniques while equation expressions are needed for unique solution of equations.

We recall Theorem 11:

▶ Theorem (Theorem 11, Section 3.4). Suppose $\mathcal{R}$ is a bisimulation up to $\succ$ and context. Then there is a guarded system of equations, of the same size as $\mathcal{R}$, with only innocuous divergences, admitting $\mathcal{R}_1$ and $\mathcal{R}_2$ as solutions.

Proof (Sketch). Suppose $\mathcal{R} = \{(P_i, Q_i)\}_{i\in I}$ is a bisimulation up to $\succ$ and context. We index all the transitions of P_i from 1 to n_i, and call the corresponding action $\mu_{i,j}$, and outcome $P'_{i,j}$: we have $\forall i, \forall j \in [1, n_i], P_i \xrightarrow{\mu_{i,j}} P'_{i,j}$.

Then, by Definition 10, for any $j \in [1, n_i]$, there exist $C_{i,j}$ and $Q'_{i,j}$ such that we have the following diagram:

$$\begin{array}{ccccccc} P_i & & & & \mathcal{R} & & Q_i \\ {\scriptstyle \mu_{i,j}}\downarrow & & & & & & \Downarrow{\scriptstyle \hat{\mu}_{i,j}} \\ P'_{i,j} & (\succ \cap (\approx \cup \mathcal{C}(\mathcal{R})))^+ & C_{i,j}[\widetilde{P}] & \mathcal{C}(\mathcal{R}) & C_{i,j}[\widetilde{Q}] & \approx & Q'_{i,j} \end{array}$$

We define the system of equations $\widetilde{X} = \widetilde{E}$ as follows:

$$\forall i \in I,\ E_i = \sum_{j=1}^{n_i} \mu_{i,j}.\, C_{i,j}[\widetilde{X}]$$

Since $\succ$ $(\xrightarrow{\tau}^+)$ terminates, $\mathcal{R} \subseteq \approx$, hence $\widetilde{P} \approx \widetilde{Q}$. We prove that $\widetilde{P}$ is solution of $\widetilde{X} = \widetilde{E}$: we have that $P_i \sim \sum_{j=1}^{n_i} \mu_{i,j}.\, P'_{i,j}$, and that $(\succ \cap (\approx \cup \mathcal{C}(\mathcal{R})))^+ \subseteq \approx$ (because $\mathcal{R} \subset \approx$). Therefore, $P'_{i,j} \approx C_{i,j}[\widetilde{P}]$, and $P_i \approx \sum_{j=1}^{n_i} \mu_{i,j}.\, C_{i,j}[\widetilde{P}]$. This shows that $\widetilde{P}$ is a solution of $\widetilde{X} = \widetilde{E}$. Since $\widetilde{P} \approx \widetilde{Q}$, $\widetilde{Q}$ is also solution.

It is left to prove that if for some i, $K_{\widetilde{E},i}$ has a non-innocuous divergence, then $\succ$ $(\xrightarrow{\tau}^+)$ does not terminate. This will lead a contradiction, and hence we will be able to apply Theorem 8 to finish the proof.

We have that $\forall i, j,\ P'_{i,j} \succ^+ C_{i,j}[\widetilde{P}]$, hence, since $\succ$ is transitive, $\forall i, j,\ P'_{i,j} \succ C_{i,j}[\widetilde{P}]$. Assume that $K_{\widetilde{E},i} \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} E_0[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau} \xrightarrow{\tau} \ldots$ (and this divergence is not innocuous).

We have that: $\widetilde{E}$ is guarded, the divergence is not innocuous, and $\forall i, P_i \sim_1 E_i[\widetilde{P}]$. This entails that there exists E'_0 such that $E_0[\widetilde{E}[\widetilde{P}]] \xrightarrow{\tau}^+ E'_0[\widetilde{P}]$, $E'_0[\widetilde{P}] \xrightarrow{\tau}$, and the latter transition is not an expression transition (meaning that $\widetilde{P}$ contributes to the transition). Assume, for simplicity, that this transition is of the form $E'_0[\widetilde{P}] \xrightarrow{\tau} E'_0[(P_j)_{j<i}, P'_{i,k}, (P_j)_{j>i}]$ for some k, with $P_i \xrightarrow{\tau} P'_{i,k}$ (i.e., E'_0 has only one instance of each variable, and this transition involves only P_i). Then $E'_0[(P_j)_{j<i}, P'_{i,k}, (P_j)_{j>i}] \succ E'_0[(P_j)_{j<i}, C_{i,k}[\widetilde{P}], (P_j)_{j>i}]$. By analysing the τ steps from $\widetilde{K}_{\widetilde{E}}$, we can continue this construction, and build an infinite sequence for $\succ$ $(\xrightarrow{\tau}^+)$. □

Note that Theorem 8 is essential to establish this completeness result, and that Theorem 6 would be insufficient.

C tyft/tyxt rule formats

In the tyft/tyxt formats, lookaheads are possible; this breaks the autonomy preservation property. Hence Theorem 17 does not hold in a tyft/tyxt language. We provide a counter-example, in a language with three constructions: first, a prefix constructor, like the prefix of CCS, and with the same rule (it is indeed a tyft rule). We assume that we can use the prefix construction with at least two distinct labels, a and b. Second, a constant $\mathbf{0}$, that has no rule (as in CCS). Third, a constructor `forward` with the following tyft rule:

$$\frac{x \xrightarrow{\mu'} y \qquad y \xrightarrow{\mu} z}{\texttt{forward}(x) \xrightarrow{\mu} z}$$

Then, the equation $X = a.\texttt{forward}(X)$ is autonomous or (strongly) guarded, but does not enjoy unique solution for $\approx$: both $a.\mathbf{0}$ and $a.b.\mathbf{0}$ are solution for $\approx$.

D $A\pi$: Operational Semantics

Processes and abstractions. For the π-calculus, we inherit the notations from CCS. Thus P, Q range over processes and small letters $a, b, \ldots, x, y, \ldots$ range over the infinite set of names. We actually consider the *asynchronous* π-calculus because its theory is simpler than that of the synchronous π-calculus; notably, bisimulation does not require closure under name instantiation.

Since the operational semantics of the π-calculus makes use of name instantiation, we impose that recursive definitions, and hence also equations, are parametrised over a set of names. We call such parametrised processes *abstractions*. The actual instantiation of the parameters of an abstraction F is done via the *application* construct $F\langle\widetilde{a}\rangle$. Processes and abstractions form the set of *agents*, ranged over by A. Substitutions are ranged over by σ.

The grammar of the calculus is thus:

$$\begin{array}{lcll} A & := & P \mid F & \text{(agents)} \\ P & := & \mathbf{0} \mid a(\widetilde{b}).P \mid \overline{a}\langle\widetilde{b}\rangle \mid P_1 \mid P_2 \mid \boldsymbol{\nu} a\, P \mid F\langle\widetilde{a}\rangle \mid\ !a(\widetilde{b}).P & \text{(processes)} \\ F & := & (\widetilde{a})\,P \mid K & \text{(abstractions)} \end{array}$$

Inputs $c(\widetilde{a}).P$ and $!c(\widetilde{a}).P$, abstractions $(\widetilde{a})\,P$, and restrictions $\boldsymbol{\nu} a\, P$ are binders for the names $\widetilde{a}$ and a, with scope P. In $(\widetilde{a})\,P$, $\widetilde{a}$ is a non-empty tuple of names. An agent is *closed* if it does not have free names. (Since the number of recursive definitions may be infinite, some care is necessary in the definition of free names of an agent, using a least fixed-point construction.) Replication could be avoided in the syntax since it can be encoded with recursion. However it is a useful construct for examples and it is therefore convenient to have it as a primitive in the syntax because some of the conditions for our unique-solution theorems will then be easier to check. Along the lines of CCS, K is used to stand for recursively defined abstractions, and each K should have a defining equation of the form $K \stackrel{\triangle}{=} F$, where F is a closed abstraction.

As for recursive definitions, so in equations the expression in the body is a closed abstraction, possibly containing equation variables. Thus in the case of the π-calculus, E ranges over abstractions and, accordingly, the solutions of equations are abstractions. As in

CCS, we can turn equations into recursive definitions; this yields the *syntactic solution* of the equations.

Since the calculus is polyadic, we assume a *sorting system* [15] to avoid disagreements in the arities of the tuples of names carried by a given name. The sorting is extended to agents in the expected manner. Similarly, when considering contexts and equation variables, it is intended that each hole and each variable has a sort so that, if they stand for an abstraction (as opposed to a process) we know the number and the sorts of their parameters. A context itself has sorts, as it may represent an abstraction or a process and, accordingly, context composition is supposed to be well-sorted. We will not present the sorting system because it is unessential. The reader should take for granted that all agents described obey a sorting.

Transitions. As in CCS, transitions are of the form $P \xrightarrow{\mu} P'$, where the grammar for actions is given by

$$\mu \quad := \quad a(\widetilde{b}) \;\mid\; \boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle \;\mid\; \tau \ .$$

$\boldsymbol{\nu}\emptyset\,\overline{a}\langle\widetilde{b}\rangle$ is written simply $\overline{a}\langle\widetilde{b}\rangle$. We define the sets of bound (resp. free) names of an action μ, written $\mathrm{bn}(\mu)$ (resp. $\mathrm{fn}(\mu)$), as follows, where $\mathrm{na}(\mu)$ stands for the set of the names of μ:

$$\mathrm{bn}(a(\widetilde{b})) = \widetilde{b} \qquad \mathrm{bn}(\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle) = \widetilde{d} \qquad \mathrm{bn}(\tau) = \emptyset \qquad \mathrm{fn}(a(\widetilde{b})) = \{a\}$$

$$\mathrm{fn}(\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle) = \{a\} \cup (\widetilde{b} \setminus \widetilde{d}) \qquad \mathrm{fn}(\tau) = \emptyset \qquad \mathrm{na}(\mu) = \mathrm{bn}(\mu) \cup \mathrm{fn}(\mu)$$

$$\frac{}{a(\widetilde{b}).\,P \xrightarrow{a(\widetilde{b})} P} \qquad \frac{}{!a(\widetilde{b}).\,P \xrightarrow{a(\widetilde{b})} !a(\widetilde{b}).\,P \mid P} \qquad \frac{}{\overline{a}\langle\widetilde{b}\rangle \xrightarrow{\overline{a}\langle\widetilde{b}\rangle} \mathbf{0}}$$

$$\frac{P \xrightarrow{\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle} P'}{\boldsymbol{\nu}n\,P \xrightarrow{\boldsymbol{\nu}(\{n\}\cup\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P'}\, n \in \widetilde{b} \qquad \frac{P \xrightarrow{\mu} P'}{\boldsymbol{\nu}n\,P \xrightarrow{\mu} \boldsymbol{\nu}n\,P'}\, d \notin \mathrm{na}(\mu) \qquad \frac{P \xrightarrow{a(\widetilde{b})} P' \qquad Q \xrightarrow{\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b'}\rangle} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}\widetilde{d}\,(P'\{\widetilde{b'}/\widetilde{b}\} \mid Q')}$$

$$\frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\, \mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset \qquad \frac{P\{\widetilde{b}/\widetilde{a}\} \xrightarrow{\mu} P'}{((\widetilde{a})\,P)\langle\widetilde{b}\rangle \xrightarrow{\mu} P'} \qquad \frac{F\langle\widetilde{a}\rangle \xrightarrow{\mu} P'}{K\langle\widetilde{a}\rangle \xrightarrow{\mu} P'}\ \text{if } K \triangleq F$$

Figure 3 $A\pi$: Labelled Transition Semantics

Figure 3 presents the transition rules for $A\pi$. We maintain from CCS the notations for transitions, such as $\Rightarrow$, $\stackrel{\widehat{\mu}}{\Longrightarrow}$, and so on.