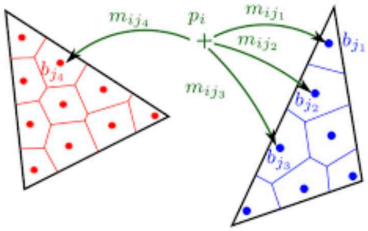
AGENCE NATIONALE DE LA RECHERCHE

ANR

REPAS

RELIABLE AND PRIVACY-AWARE SOFTWARE SYSTEMS



Deliverable D4.c

Logical Methods in Computer Science
Volume 15, Issue 3, 2019, pp. 12:1–12:34
https://lmcs.episciences.org/

Submitted Jul. 02, 2018
Published Aug. 07, 2019

DIVERGENCE AND UNIQUE SOLUTION OF EQUATIONS

ADRIEN DURIER, DANIEL HIRSCHKOFF, AND DAVIDE SANGIORGI

Univ. Lyon, ENS de Lyon, CNRS, UCB Lyon 1, LIP
e-mail address: adrien.durier@ens-lyon.fr, daniel.hirschkoff@ens-lyon.fr

INRIA and Università di Bologna
e-mail address: davide.sangiorgi@cs.unibo.it

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 use them to revisit the completeness part of the proof of full abstraction 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.

Key words and phrases: Bisimilarity, unique solution of equations, termination, process calculi.

* This is an extended version of the paper with the same title, published in the proceedings of CONCUR 2017 [DHS17].

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.23638/LMCS-15(3:12)2019

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 [Mil89]. The method is important in verification techniques and tools based on algebraic reasoning [Ros10, BBR10, GM14]. Not all equations have a unique solution: for instance any process trivially satisfies $X = X$. To ensure unique solution, it is often required that equations satisfy some kind of *guardedness* condition; see, e.g., [BW90, GM14]. In CCS, the notion of guardedness is syntactic: variables of the equations can only appear underneath a visible prefix. This is not sufficient to guarantee uniqueness. Hence, in Milner's theorem [Mil89], uniqueness of solutions relies on an additional limitation: the equations must be 'sequential', that is, the variables of the equations may not be preceded, in the syntax tree, by the parallel composition operator. 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 book about CSP [Hoa85], and plays an equally essential role in the theory of CSP.

In order to overcome such limitations, a variant of the technique, called *unique solution of contractions*, has been proposed [San15]. 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: 1. the equational theory of the contraction preorder associated to an equivalence is not the same as the equational theory of the equivalence itself, which thus needs to be studied as well; 2. the contraction preorder is strictly finer than the equivalence, hence there are equivalent processes, one of which might be solution of a given contraction, while the other is not, and the technique might not be applicable in this case.

In this paper we explore a different approach, inspired by results by Roscoe in CSP [Ros97, Ros92], essentially stating that a guarded equation (or system of equations) whose infinite unfolding never produces a divergence has the unique-solution property. Roscoe's result is presented, as usual in CSP, with respect to denotational semantics and failure based equivalence [BHR84, BR84]. 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 [Ros92] 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. Notably, we can ignore divergences that appear after finite unfoldings of the equations, called *innocuous* divergences in this paper. (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). In this abstract formulation, the body of an equation becomes a function on the states of the LTS. The theorems for CCS 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 [BIM88] (assuming appropriate hypotheses, among which congruence properties). In contrast, the analogous theorems fail for languages whose constructs follow the *tyft/tyxt* [GV92] 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 [Dur17].

The bisimulation proof method is a well-established and extensively studied technique for deriving bisimilarity results. An advantage of this method are its enhancements, provided by the so called 'up-to techniques' [PS11]. Powerful such 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 an *'up to transitivity and context'* technique due to Pous, which relies on a termination hypothesis [Pou08]. This technique generalises 'up to expansion' and combines it with 'up to context' and 'up to transitivity'. We show that, modulo an additional hypothesis, 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 (where the size of a relation is the number of its pairs, and the size of a system of equations is the number of its equations) for which uniqueness of solutions holds.

An important difference between unique solution of equations and up-to techniques arises in the (asynchronous) π-calculus. In this setting, forms of bisimulation enhancements that involve 'up to context', such as 'up to expansion and context', require closure of the candidate relation under substitutions (or instantiation of the formal parameters of an abstraction with arbitrary actual parameters). 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 call-by-name λ-calculus into the π-calculus [San00, SX18] with respect to Levy-Longo Trees (LTs). The proof in [San00, SX18] 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. (By the time the revision of this paper has been completed, we have used to the technique to establish full abstraction for the encoding of the call-by-value λ-calculus [DHS18].)

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.

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

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

Figure 1: The LTS for CCS

Comparison with the results published in [DHS17]*:* this paper is an extended version of [DHS17]. We provide here detailed proofs which were either absent or only sketched in [DHS17], notably for: our two main unique-solution theorems (Theorems 3.5 and 3.7); the comparison with up-to techniques in Section 3.4.2; the analysis of call-by-name λ-calculus in the asynchronous π-calculus (Theorem 5.12). We also include more detailed discussions along the paper, notably about the applicability of our techniques (in particular, Lemma 5.5, Propositions 4.17 and Proposition 4.14, and Remark 3.9).

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. The class of the CCS processes is built from the operators of parallel composition, guarded sum, restriction, and constants, and the guard of a sum can be an input, an output, or a silent prefix:

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

where I is a countable indexing set. Sums are guarded to ensure that behavioural equivalences and preorders are substitutive. We write $\mathbf{0}$ when I is empty, and $P+Q$ for binary sums, with the understanding that, to fit the above grammar, P and Q should be sums of prefixed terms. Each constant K has a definition $K \triangleq P$. We sometimes omit trailing $\mathbf{0}$, e.g., writing $a \mid b$ for $a.\,\mathbf{0} \mid b.\,\mathbf{0}$. We write $\mu^n.\,P$ for P preceded by n μ-prefixes. In a few examples we write $!\mu.\,P$ as abbreviation for the constant $K_{\mu.P} \triangleq \mu.\,(P \mid K_{\mu.P})$. The operational semantics is given by means of a Labelled Transition System (LTS), and is given in Figure 1 (the symmetric versions of the rules `parL` and `comL` have been omitted). The *immediate derivatives* of a process P are the elements of the set $\{P' \mid P \xrightarrow{\mu} P' \text{ for some } \mu\}$. We use ℓ to range over visible actions (i.e., inputs or outputs, excluding τ).

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 $P \stackrel{\hat{\mu}}{\Longrightarrow} P'$ holds if $P \stackrel{\mu}{\Longrightarrow} P'$ or ($\mu = \tau$ and $P = P'$). We write $P(\xrightarrow{\mu})^n P'$ if P can become P' after performing n μ-transitions. Finally, $P \xrightarrow{\mu}$ holds if there is P' with $P \xrightarrow{\mu} P'$, and similarly for other forms of transitions.

Further notations. Letters $\mathcal{R}, \mathcal{S}$ range over relations. We use the infix notation for relations, e.g., $P\ \mathcal{R}\ Q$ means that $(P,Q) \in \mathcal{R}$, and we write $\mathcal{RS}$ for the composition of $\mathcal{R}$ and $\mathcal{S}$. A relation *terminates* if there is no infinite sequence $P_1 \mathcal{R} P_2 \mathcal{R} \ldots$. 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 symbol $\stackrel{\text{def}}{=}$ for abbreviations; for instance, $P \stackrel{\text{def}}{=} G$, where G is some expression, means that P stands for the expression G. In contrast, symbol $\stackrel{\triangle}{=}$ is used for the definition of constants, and $=$ for syntactic equality and for equations. If $\leq$ is a preorder, then $\geq$ is its inverse (and conversely).

We focus on *weak* behavioural relations, which abstract from the number of internal steps performed by equivalent processes, because they are, in practice, the important relations.

Definition 2.1 (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. 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 2.2. 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.)

In the equation $X = E[X]$, we sometimes call body of the equation the equation expression E (we use the same terminology for systems of equations).

$E[\widetilde{P}]$ 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 2.3. 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 $\widetilde{P} \approx \widetilde{Q}$.

Examples of systems with a unique solution for $\approx$ are:

(1) $X = a.X$
(2) $X_1 = a.X_2$, $X_2 = b.X_1$

The unique solution of the system (1), modulo $\approx$, is the constant $K \stackrel{\triangle}{=} a.K$: for any other solution P we have $P \approx K$. The unique solution of (2), modulo $\approx$, is given by the constants K_1, K_2 with $K_1 \stackrel{\triangle}{=} a.K_2$ and $K_2 \stackrel{\triangle}{=} b.K_1$; again, for any other pair of solutions P_1, P_2 we have $K_1 \approx P_1$ and $K_2 \approx P_2$.

Examples of systems that do not have unique solution are:

(1) $X = X$
(2) $X = \tau.\, X$
(3) $X = a \mid X$

All processes are solutions of (1) and (2); examples of solutions for (3) are K and $K \mid b$, for $K \stackrel{\triangle}{=} a.\, K$

Remark 2.4. To prove that two processes P and Q are equivalent using the unique solution proof technique, one has first to find an equation $X = E[X]$ of which both P and Q are solutions. Then, a sufficient condition for uniqueness of solutions makes it possible to deduce $P \approx Q$.

The unique-solution method is currently particularly used in combination with algebraic laws [Ros10, BBR10, GM14].

Definition 2.5. An equation expression E is

- *strongly guarded* if each occurrence of a variable in E is underneath a visible prefix;
- *(weakly) guarded* if each occurrence of a variable in E is underneath a prefix, visible or not;
- *sequential* if each occurrence of a variable in E is only appears underneath prefixes and sums .

We say that an equation satisfies one of the above properties when its body does. These notions are extended to systems of equations in a natural way: for instance, $\{X_i = E_i\}_{i \in I}$ is guarded if each expression E_i is (w.r.t. every variable that occurs in E_i).

In other words, if the system is sequential, then for every expression E_i, any sub-expression of E_i in which X_j appears, apart from X_j itself, is a sum (of prefixed terms). For instance,

- $X = \tau.\, X + \mu.\, \mathbf{0}$ is sequential but not strongly guarded, because the guarding prefix for the variable is not visible.
- $X = \ell.\, X \mid P$ is guarded but not sequential.
- $X = \ell.\, X + \tau.\, \boldsymbol{\nu} a\, (a.\, \overline{b} \mid a.\, \mathbf{0})$, as well as $X = \tau.\, (a.\, X + \tau.\, b.\, X + \tau)$ are both guarded and sequential.

Remark 2.6 (Recursive specifications in ACP)**.** Systems of equations are called *recursive specifications* in the literature related to ACP [BW90]. In that context, other notions of guardedness have been studied. A sufficient condition, more powerful than Milner's, was originally given by Baeten, Bergstra and Klop [BW90], where synchronisation is transformed into a visible action, which is then deleted through an explicit operator. An equation is then guarded if no such deletion operator appears in its body.

In some process calculi, such as ACP [BW90] and mCRL2 [GM14], guardedness is synonymous, for an equation, to having a unique solution: this follows the *Recursive specification principle* (RSP), which states that guarded recursive specifications are unique (while the *Recursive definition principle* states that recursive specifications do have solutions). Increasingly general definitions of guardedness that make this principle sound are then studied. In this framework, our contribution can be seen as a new notion of guardedness, stronger that what is found in the literature, under which the recursive definition principle is sound for weak bisimilarity.

Theorem 2.7 (unique solution of equations, [Mil89])**.** *A system of strongly guarded and sequential equations has a unique solution for* $\approx$.

The proof exploits an invariance property on immediate transitions for strongly guarded and sequential expressions, and then extracts a bisimulation (up to bisimilarity) out of the solutions of the system.

To see the need of the sequentiality condition, consider the equation (from [Mil89])

$$X = \boldsymbol{\nu} a\,(a.\,X \mid \overline{a})\ ,$$

where the occurrence of X in the equation expression is strongly guarded but not sequential. Any process that does not use a is a solution.

Note. In CCS, ν is not a binder; this allows us to write equations where local names are used outside their scopes (for instance, $X = a.\,\boldsymbol{\nu} a\,(\overline{a} \mid X)$), as there is no alpha-renaming. It would still be possible, when ν has to be considered a binder (in a higher-order setting, for example), to write similar equations in a parametric fashion: $X = (a)\ a.\,\boldsymbol{\nu} a\,(\overline{a} \mid X\langle a\rangle)$. Such an approach is adopted in Section 5, to handle name passing.

Definition 2.8 (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 2.9. 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 $X = E$: we define the n-th unfolding of E to be E^n; thus E^1 is defined as E, E^2 as $E[E]$, and E^{n+1} as $E^n[E]$. 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.*

For a system of equations $\widetilde{X} = \widetilde{E}[\widetilde{X}]$, the unfoldings are defined accordingly (where E_i replaces X_i in the unfolding): we write $\widetilde{E}^2$ for the system $\{X_i = E_i[\widetilde{E}]\}_{i\in I}$, and similarly for $\widetilde{E}^n$. The syntactic solutions are defined to be the set of mutually recursive constants $\{K_{\widetilde{E},i} \stackrel{\triangle}{=} E_i[\widetilde{K}_{\widetilde{E}}]\}_i$.

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 $E[\widetilde{P}]$, where the transition emanates from the E component only:

Lemma 3.1 (Expression transitions)**.**

(1) *Given E and E' two equation expressions, if $E \xrightarrow{\mu} E'$, then $E[\widetilde{P}] \xrightarrow{\mu} E'[\widetilde{P}]$, for all processes $\widetilde{P}$, and $E[\widetilde{F}] \xrightarrow{\mu} E'[\widetilde{F}]$ for all equation expressions $\widetilde{F}$.*

(2) *If E is a guarded expression and $E[\widetilde{P}] \xrightarrow{\mu} T$, then there is an expression E' such that $E \xrightarrow{\mu} E'$ and $T = E'[\widetilde{P}]$. Similarly for a transition $E[\widetilde{F}] \xrightarrow{\mu} E'$.*

Proof. (1) By a simple induction on the derivation of the transition.
(2) By a simple induction on the equation expression E.

□

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

Definition 3.2 (Reducts)**.** (1) The *set of reducts* of an expression E, written $\mathtt{red}(E)$, is given by:

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

(2) The set of *reducts of the unfoldings* of a system of equations $\{X_i = E_i[\widetilde{X}]\}_{i \in I}$, also written $\mathtt{red}_\omega(\widetilde{E})$, is defined as

$$\mathtt{red}_\omega(\widetilde{E}) \quad \triangleq \quad \bigcup_{n \in \mathbb{N}, i \in I} \mathtt{red}(E_i^n)$$

Definition 3.3. A system of equations $\widetilde{E}$ *protects its solutions* if, for all solution $\widetilde{P}$ of the equation $\widetilde{X} = \widetilde{E}[\widetilde{X}]$, the following holds: for all $E' \in \mathtt{red}_\omega(\widetilde{E})$, if $E'[\widetilde{P}] \stackrel{\mu}{\Rightarrow} Q$ for some μ and Q, then there exists E'' and n such that $E'[\widetilde{E}^n] \stackrel{\hat{\mu}}{\Rightarrow} E''$, and $E''[\widetilde{P}] \approx Q$.

Consider a single equation $X = E[X]$: it 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 is a technical condition, which is useful in the proofs. In this paper, all examples of equations having a unique solution satisfy this property.

Proposition 3.4. *A system of equations that protects its solutions has a unique solution for $\approx$.*

Proof. Given two solutions $\widetilde{P}$, $\widetilde{Q}$ of the system of equations $\widetilde{X} = \widetilde{E}[\widetilde{X}]$ we prove that the relation

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

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

We consider $(S,T) \in \mathcal{R}$, that is, $S \approx E'[\widetilde{P}]$ and $T \approx E'[\widetilde{Q}]$, for some $E' \in \mathtt{red}_\omega(\widetilde{E})$. If $S \xrightarrow{\mu} S'$, then by bisimilarity $E'[\widetilde{P}] \stackrel{\hat{\mu}}{\Rightarrow} S'' \approx S'$. Since $\widetilde{E}$ protects its solutions, there are n, E'' such that $E'[\widetilde{E}^n] \stackrel{\hat{\mu}}{\Rightarrow} E''$ and $E''[\widetilde{P}] \approx S''$. We can deduce by Lemma 3.1(1) that $E'[\widetilde{E}^n[\widetilde{Q}]] \stackrel{\hat{\mu}}{\Rightarrow} E''[\widetilde{Q}]$. Since $\widetilde{Q}$ is a solution of $\widetilde{E}$, we have $\widetilde{Q} \approx \widetilde{E}^n[\widetilde{Q}]$. This entails, as we know by hypothesis $T \approx E'[\widetilde{Q}]$, that $T \approx E'[\widetilde{Q}] \approx E'[\widetilde{E}^n[\widetilde{Q}]]$. By bisimilarity, we deduce from $E'[\widetilde{E}^n[\widetilde{Q}]] \stackrel{\hat{\mu}}{\Rightarrow} E''[\widetilde{Q}]$ that $T \stackrel{\hat{\mu}}{\Rightarrow} T' \approx E''[\widetilde{Q}]$.

The situation can be depicted on the following diagram:

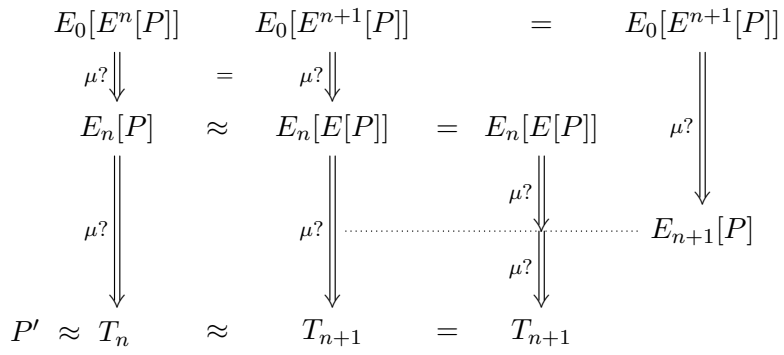
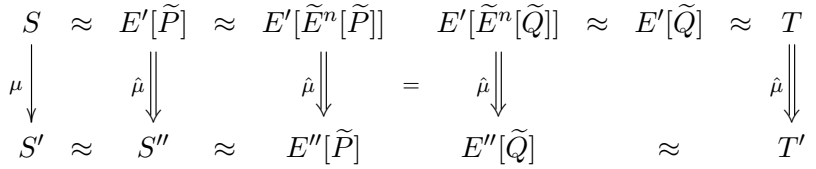


Figure 2: Recursion: construction of the sequence of transitions $E_0[E^n[P]] \stackrel{\mu?}{\Longrightarrow} E_n$



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

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

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.

Theorem 3.5 (Unique solution)**.** *A guarded system of equations whose syntactic solutions do not diverge has a unique solution for* $\approx$*.*

Proof. To enhance readability, we first give the proof for a single equation $X = E[X]$. We discuss the generalisation to a system of equations at the end of the proof.

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 $E_0[E^{n+1}] \stackrel{\mu?}{\Longrightarrow} E_{n+1}$

This construction is illustrated by Figure 3. We start with the empty sequence from E_0.

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

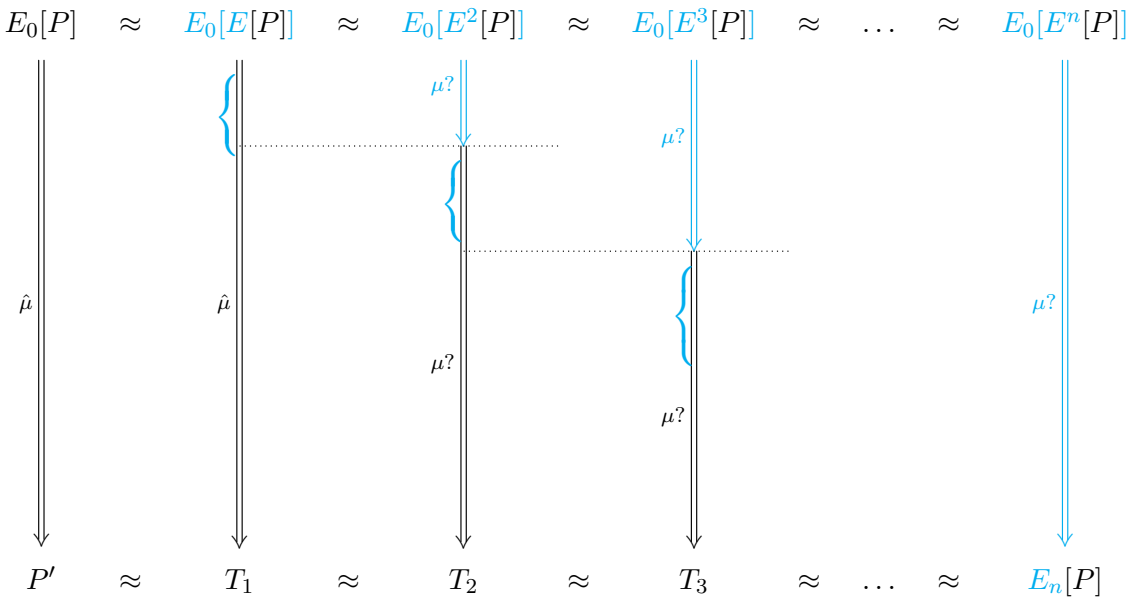


Figure 3: Construction of the expression transition

- 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 further equation E: we have $E_0[E^{n+1}] \stackrel{\mu}{\Rightarrow} E_n[E]$ by Lemma 3.1. 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.

Systems of equations. To extend the previous proof to systems of equations, we consider solutions $\widetilde{P}$ to the system $\widetilde{E}$, and use unfoldings of the system of equations (instead of unfoldings of a single equation). We also reason about an initial transition from $E_0[\widetilde{P}]$, where E_0 is a reduct of the unfoldings of one of the equations (i.e., $E_0 \in \mathtt{red}_\omega(\widetilde{E})$). □

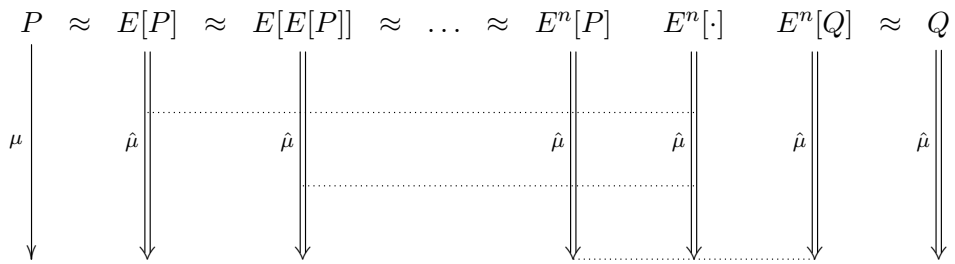


Figure 4: Proof of Theorem 3.5, building a a transition of $E^n[Q]$ where Q does not move

3.2. **Innocuous Divergences.** In the remainder of the section we refine Theorem 3.5 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, a finite unfolding may produce the same divergence, therefore 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. Those divergences can be understood as instances, in the syntactic solution, of divergences of some finite unfolding of the equation: Consider a (single) equation $X = E[X]$; this way, any divergence of E^n can be transformed into an innocuous divergence of K_E. This idea is also behind Lemma 4.8 below.

This refinement fits well the operational approach we adopt to formalise the results; it looks less natural in a denotational or trace-based setting for CSP like in [Ros92, Ros97], where any divergence causes a process to be considered undefined.

Definition 3.6 (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, summing up all usages of rule `const` with one of the $K_{\widetilde{E},j}$s (including $j = i$) *in all derivation proofs of the transitions belonging to the divergence*, we obtain a finite number.

Theorem 3.7 (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 $\approx$.*

Proof. We reason like in the proof of Theorem 3.5. We explain the difference, in the case where we have a single equation (bearing in mind that the generalisation discussed at the end of the proof of Theorem 3.5 can be carried over accordingly).

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. □

Remark 3.8. The conditions for unique solution in Theorems 3.5 and 3.7 combine 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 imposing guardedness of the equations could be dropped. The resulting theorems would actually be more powerful because they would accept equations which are not syntactically guarded: it is sufficient that each equation has a finite unfolding which 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.

Remark 3.9. Even taking innocuous divergences into account, our criterion for equations to have a unique solution is not complete. Indeed, the following equation

$$X = a.X + \overline{a}.X + d.(X \mid X)$$

has a non-innocuous divergence: its syntactic solution, K, has the transition $K \xrightarrow{d} K \mid K$; then, $K \mid K$ diverges by unfolding on both sides at every step. However, the equation has a unique solution, intuitively because the prefix d acts as a strong guard, that does not take part in the divergence. This could hint at further developments of our theory.

Note moreover that this equation is non-linear, in the sense that there are occurrences of the same variable X in parallel. In practice, such examples are not very useful; for linear equations, we were not able to find counter-examples.

3.3. **Conditions for unique solution.** The following lemma states a condition to ensure that all divergences produced by a system of equations are innocuous. This condition is decidable, but weak. However, in practice, it is often satisfied; it is in particular sufficient in all examples in the paper.

Lemma 3.10. *Consider a system of equations $\widetilde{X} = \widetilde{E}$, and suppose that 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.*

Proof. The condition ensures that at least one of the prefix occurring above the equation variables may never take part in a τ action. This entails that this prefix cannot be triggered along an infinite sequence of τ steps. Hence a divergence of the unique solution of such a system of equations may not require the rule `const` to be used an infinite number of times. □

3.4. **Comparison with other techniques.** We now compare our technique with two alternative approaches: unique solution of contractions, and enhancements of bisimulation.

3.4.1. *An example (lazy and eager servers), and comparison with contractions.* We now show an example of application of our technique, taken from [San15]. The example also illustrates the relative strengths of the two unique solution theorems (Theorems 3.5 and 3.7), and a few aspects of the comparison with other bisimulation techniques.

For the sake of readability, we use a version of CCS with value passing; this could be translated into pure CCS [Mil89]. 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.

The paper [San15] presents two proofs of this equivalence. One proof makes use of the 'bisimulation up-to expansion and context' technique; this makes it possible to carry out a proof using a single pair of processes for each integer n. A proof of similar size uses the technique of 'unique solution of contractions', by establishing, with the help of a few simple algebraic laws, that $\{LS\langle n\rangle\}_n$ and $\{ES\langle n\rangle\}_n$ are solutions of the same system of contractions.

We can also build a proof using the technique of 'unique solution of equations'. Milner's original version of this technique cannot be used, because the equations make use of operators other than just prefix and sum. In contrast, Theorem 3.5 can be applied. 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. It is essentially the same proof as that for unique solution of contractions.

To apply Theorem 3.5, we also need to check that the equations do 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 on the other hand the hypothesis that R may not diverge is lifted, then Theorem 3.5 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 3.7.

We can relax the definition of R even further and allow calls back to the main server from R itself. In this case, interactions between R and the main server (eager or lazy) may yield divergences (for instance setting $R \stackrel{\text{def}}{=} \overline{c}$). Such divergences need not be innocuous, as intuitively they require infinitely many unfolding of the body of an equation. Thus now even Theorem 3.7 is not applicable. (More precisely, for Theorem 3.7 to fail $R\langle n, z\rangle$, for each n and z, should have the possibility of performing an output at c as first visible transition.)

This becomes therefore an example in which the 'unique solution of contraction' technique is more powerful, as such technique does not rely on conditions about divergence and is therefore applicable.

3.4.2. *Comparison with up-to 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 [San15]; for instance it cannot handle the server example of Section 3.4.1.

The technique of 'unique solution of contractions' [San15] relies on the theory of an auxiliary preorder (contraction), needed to establish the meaning of 'solution'; and the soundness theorems in [San15] use a purely syntactic condition (guarded variables). In contrast, our techniques with equations do not rely on auxiliary relations and their theory, but the soundness theorems use a semantic condition (divergence) , see also Remark 3.8).

The two techniques are incomparable. Considering the server example of Section 3.4.1, 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, while two bisimilar processes are solutions of exactly the same set of equations, they need not be solutions of the same contractions. For instance, 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 solutions of the equation $X = a.\, X$; in contrast, only H is a solution 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 [Pou08]. This technique allows one 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 [PS11], here they are combined safely thanks to a 'control relation', written below $\succ$, which satisfies a termination hypothesis. This control relation is used to make sure there is no cyclic or infinite sequence of τ transitions used to hide other potential visible transitions. In that regard, this condition is very similar to the non-divergence hypothesis of Theorem 3.7, as can be seen in the proofs below.

Formally, contexts are processes that can contain holes, indexed by some finite set of integers. Therefore, contexts 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, keeping in mind that contexts are needed to study up-to techniques while equation expressions are needed for unique solution of equations.

We write $\mathcal{C}(\mathcal{R})$ for the context closure of a relation $\mathcal{R}$, defined as 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 3.11. Let $\succ$ be a relation that is transitive, closed under contexts, and such that $\succ$ $(\xrightarrow{\tau}^+)$ terminates. A relation $\mathcal{R}$ is a *bisimulation up to* $\succ$ *and context* if, whenever $P\ \mathcal{R}\ Q$:

(1) if $P \xrightarrow{\mu} P'$ 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.

We refer to [Pou08] for more details on this up-to technique, and for the the proof of its soundness.

We remark that in [Pou08], instead of the transitive closure $(\succ \cap \overline{\mathcal{R}})^+$ in Definition 3.11, we have a *reflexive* and transitive closure. We do not know if relaxing this technical condition breaks Theorem 3.12 below.

We introduce some notations in order to state the correspondence between the technique introduced in Definition 3.11 and systems of equations.

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 $(i = 1, 2)$.

In the following statement, the *size* of a relation is the number of pairs it contains, and the size of a system of equations is the number of equations it consists of.

Theorem 3.12 (Completeness with respect to up-to techniques)**.** *Suppose $\mathcal{R}$ is a bisimulation up to $\succ$ and context. Then there exists a guarded system of equations, with only innocuous divergences, that admits $\mathcal{R}_1$ and $\mathcal{R}_2$ as solutions. Moreover, this system has the same size as $\mathcal{R}$.*

Proof. Suppose $\mathcal{R} = \{(P_i, Q_i)\}_{i\in I}$ is a bisimulation up to $\succ$ and context.

We index the transitions of P_i from 1 to n_i, and write $P_i \xrightarrow{\mu_{i,j}} P'_{i,j}$ for $1 \le j \le n_i$.

Then, by Definition 3.11, for $1 \le j \le n_i$, there exist $C_{i,j}$ and $Q'_{i,j}$ such that we have the following diagram:

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

We define the system of equations $\widetilde{X} = \widetilde{E}$ by setting $\forall i \in I,\ E_i = \sum_{j=1}^{n_i} \mu_{i,j}.\, C_{i,j}[\widetilde{X}]$. By construction, this system and $\mathcal{R}$ have the same size.

We prove that $\widetilde{P}$ is a solution of $\widetilde{X} = \widetilde{E}$. We have that $P_i \sim \sum_{j=1}^{n_i} \mu_{i,j}.\, P'_{i,j}$. Moreover, by the results in [Pou08], since $\succ$ $(\xrightarrow{\tau}^+)$ terminates, $\mathcal{R} \subseteq \approx$, which implies $(\succ \cap (\approx \cup \mathcal{C}(\mathcal{R})))^+ \subseteq \approx$. 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 $\mathcal{R} \subseteq \approx$, we have $\widetilde{P} \approx \widetilde{Q}$, hence $\widetilde{Q}$ is also a solution of the system of equations.

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. As this would be contradictory, we will be able to apply Theorem 3.7 to finish the proof.

Assume that $K_{\widetilde{E},k} \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} F_0[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau}\xrightarrow{\tau} \ldots$, and that this divergence is not innocuous. Based on this divergence, we build a sequence of equation expressions F_n and F'_n such that:

$$F_0[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau}{}^* F'_0[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau} F_1[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau}{}^* F'_1[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau} F_2[\widetilde{K}_{\widetilde{E}}] \ldots$$

In the above divergence, we have that for all n, $F_n \xrightarrow{\tau}{}^* F'_n$ (these are expression transitions) and $F'_n[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau} F_{n+1}[\widetilde{K}_{\widetilde{E}}]$; moreover, the latter transition is not an instance of an expression transition from F'_n (meaning that $\widetilde{K}_{\widetilde{E}}$ contributes to the transition). We will then show that $F'_n[\widetilde{P}] \xrightarrow{\tau}\succ F_{n+1}[\widetilde{P}]$. Therefore $\succ (\xrightarrow{\tau}{}^+)$ does not terminate.

F_0 is already given. Assume then that we have F_n such that there is a non innocuous divergence from $F_n[\widetilde{K}_{\widetilde{E}}]$. This entails that there exists F'_n such that $F_n[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau}{}^* F'_n[\widetilde{K}_{\widetilde{E}}]$, $F'_n[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau}$, and the latter transition is not an instance of an expression transition from F'_n. Without loss of generality, we can assume that the transitions in $F_n[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau}{}^* F'_n[\widetilde{K}_{\widetilde{E}}]$ are instances of expression transitions.

The fact that transition $F'_n[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau}$ is not an instance of an expression transition means that at least one equation expression E_i is involved in it. For readability, we assume that this transition is of the form $F'_n[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau} F'_n[(K_{\widetilde{E},k})_{k<i}, Q, (K_{\widetilde{E},k})_{k>i}]$ for some k and some Q, with $K_{\widetilde{E},i} \xrightarrow{\tau} Q$ (i.e., in F'_n there is only one copy of $K_{\widetilde{E},i}$, and the transition involves only $K_{\widetilde{E},i}$). The reasoning below can be adapted to cases where the τ-transition involves a more complex synchronisation, and/or several copies of $K_{\widetilde{E},i}$.

Since $E_i = \sum_j \mu_{i,j}.\, C_{i,j}[\widetilde{X}]$, there is j such that $\tau = \mu_{i,j}$ and $Q = C_{i,j}[\widetilde{K}_{\widetilde{E}}]$. We then fix F_{n+1} to be $F'_n\{C_{i,j}[\widetilde{X}]/X_i\}$. Indeed we have $F'_n[\widetilde{K}_{\widetilde{E}}] \xrightarrow{\tau} F_{n+1}[\widetilde{K}_{\widetilde{E}}]$.

We have observed that $P_i \sim \sum_j \mu_{i,j}.\, P'_{i,j}$, and $E_i = \sum_j \mu_{i,j}.\, C_{i,j}[\widetilde{X}]$, hence any transition from E_i can be mimicked by a transition of P_i. Therefore we have that $P_i \xrightarrow{\tau} P'_{i,j}$ and $F'_n[\widetilde{P}] \xrightarrow{\tau} F'_n[(P_k)_{k<i}, P'_{i,j}, (P_k)_{k>i}]$. We have that $P'_{i,j} \succ^+ C_{i,j}[\widetilde{P}]$ by construction, hence, since $\succ$ is transitive, $P'_{i,j} \succ C_{i,j}[\widetilde{P}]$; finally, $\succ$ is closed by contexts, hence

$$F'_n[(P_k)_{k<i}, P'_{i,j}, (P_k)_{k>i}] \succ F'_n[(P_k)_{k<i}, C_{i,j}[\widetilde{P}], (P_k)_{k>i}] = F_{n+1}[\widetilde{P}] \ .$$

The sequence of equation expressions F_n and F'_n has thus been defined in such a way that $F_n \xrightarrow{\tau}{}^* F'_n$, hence $F_n[\widetilde{P}] \xrightarrow{\tau}{}^* F'_n[\widetilde{P}]$. We also have $F'_n[\widetilde{P}] \xrightarrow{\tau}\succ F_{n+1}[\widetilde{P}]$, thus $F_n[\widetilde{P}](\xrightarrow{\tau}{}^+) \succ F_{n+1}[\widetilde{P}]$. This is an infinite sequence for $(\xrightarrow{\tau}{}^+) \succ$, hence we also have non-termination for $\succ (\xrightarrow{\tau}{}^+)$. This yields a contradiction. □

Theorem 3.7 is essential to establish Theorem 3.12; Theorem 3.5 would be insufficient.

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 4.7, have been formalised in Coq theorem prover [Dur17].

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. As usual, $f \circ g$ denotes the composition of f and g. The CCS processes of Section 2 correspond here to the states of an LTS. In turn, a CCS context C corresponds to a state function, mapping a process P to the process $C[P]$.

Definition 4.1. A state function f *respects a relation* R whenever xRy implies $f(x)Rf(y)$, for all states x, y.

Definition 4.2 (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 functions $f_i \in \mathcal{F}$ such that for all x it holds that: $f(x) \xrightarrow{\mu_i} f_i(x)$, 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 transitions correspond to expression transitions in CCS, and autonomous functions correspond to guarded contexts, which do not need the contribution of 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*. Such operators are defined relative to a behavioural equivalence or preorder; in this section we are interested in bisimilarity, hence the relation R in the definition below should be understood to be $\approx$.

Definition 4.3 (Set of operators)**.** Consider an LTS $\mathcal{T}$, a binary relation R on $\mathcal{T}$, and a set $\mathcal{O}$ of functions from S to S. We say that $\mathcal{O}$ is a *set of R-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) R is preserved, that is, all functions in $\mathcal{O}$ respect R

A 'symmetric variant' of clause 3 always holds: if f is autonomous, then so is $f \circ g$: indeed, transitions of $f(x)$ do not depend on x, hence transitions of $f(g(x))$ do not depend on either x or g. Clause 4 expresses congruence of the equivalence w.r.t. the set of contexts: here, the state functions in $\mathcal{O}$. The autonomous transitions of a set of R-operators yield an LTS whose states are the operators themselves. Such transitions are of the form $f \xrightarrow{\mu} g$.

In the remainder of this section, we assume R to be equal to bisimilarity, and call *set of operators* any set of $\approx$-operators, without specifying the relation. Where the underlying set $\mathcal{O}$ of operators is clear, we simply call a function belonging to $\mathcal{O}$ an *operator*.

We define weak transitions as follows: $f \Rightarrow g$ if there is a sequence of autonomous transitions $f \xrightarrow{\tau} f_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} f_n \xrightarrow{\tau} g$. $\stackrel{\mu}{\Longrightarrow}$ is then simply $\Rightarrow\xrightarrow{\mu}\Rightarrow$.

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, using n-ary functions).

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:

$$F := f \circ F \qquad \text{for } f \in \mathcal{O}$$

(Note that this syntax only deals with infinite compositions. Since $\mathcal{O}$ is closed under finite compositions, we do not need to handle such compositions in the above definition.) In particular, we write f^∞ for the infinite term $f \circ f \circ f \circ \ldots$.

We define the autonomous transitions for such infinite terms using 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.

These rules are consistent for finite compositions of functions in $\mathcal{O}$, in the sense that they allow one to infer the correct transitions for finite compositions of functions.

The following lemmas show that infinite terms have the same transitions $\xrightarrow{\mu}$ and weak transitions $\stackrel{\mu}{\Longrightarrow}$ as their counterpart finite unfoldings. To build a transition $f^\infty \xrightarrow{\mu} F$ from a transition of an unfolding $f^n \xrightarrow{\mu} g$, 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 4.4. *Let f be a well-behaved operator and g an operator in some set $\mathcal{O}$. Then $g \circ f^\infty \xrightarrow{\mu} F$ for some F if and only if there is n such that $g \circ f^n \xrightarrow{\mu} h$ for some h; in which case, there is $h' =_f h$ such that $F = h' \circ f^\infty$.*

Proof.($\Rightarrow$) Assume $g \circ f^\infty \xrightarrow{\mu} F$. By a simple induction over the derivation of this transition, we get that there is n such that f^n is autonomous, $f^n \xrightarrow{\mu} h$, and $F = h \circ f^\infty$

($\Leftarrow$) Assume $g \circ f^n \xrightarrow{\mu} h$. Since f is well-behaved and composition respects autonomy, there is m such that $g \circ f^m$ is autonomous; we take that m to be the minimum among the exponents that make $g \circ f^m$ autonomous. If $n \leq m$, there is an autonomous transition $g \circ f^m \xrightarrow{\mu} h \circ (f^{m-n})$. If $m \leq n$, given that $g \circ f^m$ is autonomous, there is a transition $g \circ f^m \xrightarrow{\mu} h'$, where $h' \circ f^{n-m} = h$. Either way, $g \circ f^m \xrightarrow{\mu} h_0$, where $h_0 =_f h$.

Since m is the smallest exponent that makes $g \circ f^m$ autonomous, we can derive $(g \circ f^m) \circ f^\infty \xrightarrow{\mu} g_0 \circ f^\infty$. This concludes the proof. □

We now adapt Lemma 4.4 to weak transitions.

Lemma 4.5. *Let f be a well-behaved operator and g an operator in some set $\mathcal{O}$. Then $g \circ f^\infty \stackrel{\mu}{\Longrightarrow} F$ for some F if and only if there is n such that $g \circ f^n \stackrel{\mu}{\Longrightarrow} h$ for some h; in which case, there is $h' =_f h$ such that $F = h' \circ f^\infty$.*

Proof. For each implication, proceed by induction over the length of $\stackrel{\mu}{\Longrightarrow}$, and use Lemma 4.4. □

Definition 4.6 (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, 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 3.5.

Theorem 4.7 (Unique solution, abstract formulation)**.** *Let f be a well-behaved operator on $\mathcal{O}$ on some LTS $\mathcal{T}$. If f^∞ does not diverge, then the equation $X = f(X)$ either has no solution or has a unique solution for $\approx$.*

Proof. The proof is essentially the same as that of Theorem 3.5, replacing equations expressions with operators, from a fixed set of operators $\mathcal{O}$ (one still has to consider reducts from an operator), and replacing instantiation of equation expression transitions with instantiation of autonomous transitions. A guarded equation expression becomes an autonomous operator. □

The equation in the statement of the theorem might have no solution at all. For example, consider the LTS $(\mathbb{N}, \{a\}, \to)$ 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). The 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 4.7 can be refined along the lines of Theorem 3.7. 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.

Lemma 4.8. *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^∞: $f^\infty \xrightarrow{\mu_1} g_1 \circ f^\infty \xrightarrow{\mu_2} \ldots \xrightarrow{\mu_i} g_i \circ f^\infty \xrightarrow{\tau} g_{i+1} \circ f^\infty \xrightarrow{\tau} \ldots$ such that for all $i \geq 1$, g_i is an operator and $g_i =_f f_i$.

Proof. The proof is a simple induction on the sequence of transitions defining the divergence. At each step, we unfold f^∞ as many times as needed. □

Given a divergence Δ of f^n, we write Δ^∞ to indicate the divergence of f^∞ obtained from Δ as in Lemma 4.8. We call a divergence of f^∞ *innocuous* when it can be described in this way, that is, as a divergence Δ^∞ obtained from a divergence Δ of f^n, for some n.

Theorem 4.9 (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 $\approx$.*

Proof. Just as in CCS, where the proof of Theorem 3.5 has to be modified for Theorem 3.7, here the proof of Theorem 4.7 is to be modified. The modification is essentially the same as in CCS, again substituting equation expressions for operators. □

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 call a finite sequence of actions $s = \mu_1, \ldots, \mu_n$, where each μ_i is a visible action, a *trace*. Accordingly, an infinite sequence of such actions $s = (\mu_i)_{i\in\mathbb{N}}$ is called an *infinite trace*. Given $s = \mu_1, \ldots, \mu_n$ a trace, we write $x \stackrel{s}{\Longrightarrow}$ if $x \stackrel{\mu_1}{\Longrightarrow} x_1 \stackrel{\mu_2}{\Longrightarrow} x_2 \ldots x_{n-1} \stackrel{\mu_n}{\Longrightarrow} x_n$, for some states $x_1, \ldots, x_n$. Likewise, given $s = (\mu_i)_{i\in\mathbb{N}}$ an infinite trace, we write $x \stackrel{s}{\Longrightarrow}$ if there are states $(x_i)_{i\in\mathbb{N}}$ such that $x = x_0$ and for all $i \in \mathbb{N}$, $x_i \stackrel{\mu_i}{\Longrightarrow} x_{i+1}$.

Definition 4.10 (Trace-based relations)**.** Two states x, y are in the *trace inclusion*, written $x \subseteq_{tr} y$, if $x \stackrel{s}{\Longrightarrow}$ implies $y \stackrel{s}{\Longrightarrow}$, for each trace s. They are *trace equivalent*, written $x \approx_{tr} y$, if both $x \subseteq_{tr} y$ and $y \subseteq_{tr} x$ hold.

Two states x, y are in the *infinite trace inclusion*, written $x \subseteq_{tr^\infty} y$, if $x \stackrel{s}{\Longrightarrow}$ implies $y \stackrel{s}{\Longrightarrow}$, for each finite or infinite trace s. They are *infinite trace equivalent*, written $x \approx^\infty_{tr} y$, if both $x \subseteq_{tr^\infty} y$ and $y \subseteq_{tr^\infty} x$ hold.

The definitions from Section 4.1 are the same as for $\approx$; however we now consider sets of $\subseteq_{tr}$-operators, i.e., we are interested in operators that respect $\subseteq_{tr}$. Indeed the preorder $\subseteq_{tr}$ is used in the proof of unique solution for $\approx_{tr}$, hence operators must respect $\subseteq_{tr}$ rather than $\approx_{tr}$.

The notion of trace is extended to operators like we do for weak transitions: we write $f \stackrel{s}{\Longrightarrow}$ if there is a sequence of autonomous transitions $f \stackrel{\mu_1}{\Longrightarrow} f_1 \stackrel{\mu_2}{\Longrightarrow} \cdots \stackrel{\mu_n}{\Longrightarrow} f_n$ $(s = \mu_1, \ldots, \mu_n)$, and likewise for infinite traces.

Lemma 4.11. *Given a well-behaved $\subseteq_{tr}$-operator f, a $\subseteq_{tr}$-operator g, and a (finite) trace s, we have $g \circ f^\infty \stackrel{s}{\Longrightarrow}$ if and only if there is n such that $g \circ f^n \stackrel{s}{\Longrightarrow}$.*

Proof. We proceed by induction over the length of s, and apply Lemma 4.5 in each case. □

All theorems obtained for $\approx$ can be adapted to $\approx_{tr}$, with similar proofs. As an example, Theorem 4.9 becomes:

Theorem 4.12. *Let $f \in \mathcal{O}$ be a well-behaved $\subseteq_{tr}$-operator. If all divergences of f^∞ are innocuous, then the equation $X = f(X)$ either has no solution or has a unique solution for $\approx_{tr}$.*

Proof. We proceed by showing that $x \subseteq_{tr} f(x)$ implies $x \subseteq_{tr} f^\infty$, and then that $f(x) \subseteq_{tr} x$ implies $f^\infty \subseteq_{tr} x$. This indeed gives that $x \approx_{tr} f(x)$ implies $x \approx_{tr} f^\infty$; hence the equation has at most one solution (f^∞ does not belong to the LTS, hence it is not a solution).

(1) $f(x) \subseteq_{tr} x$ implies $f^\infty \subseteq_{tr} x$. For this part, the absence of divergence hypothesis is not needed.

Let s be a trace, and assume $f^\infty \overset{s}{\Longrightarrow}$. By Lemma 4.11, there is n such that $f^n \overset{s}{\Longrightarrow}$. Hence, $f^n(x) \overset{s}{\Longrightarrow}$. Since $f(x) \subseteq_{tr} x$ and f is an $\subseteq_{tr}$-operator, we have that $f^n(x) \subseteq_{tr} f^{n-1}(x) \subseteq_{tr} \ldots \subseteq_{tr} x$. Hence, $x \overset{s}{\Longrightarrow}$, and $f^\infty \subseteq_{tr} x$.

(2) $x \subseteq_{tr} f(x)$ implies $x \subseteq_{tr} f^\infty$. This part of the proof is very similar to the proofs of Theorems 3.7 and 4.9.

Assume $x \subseteq_{tr} f(x)$, and $x \overset{s}{\Longrightarrow}$. We want to show that there exists some n such that $f^n \overset{s}{\Longrightarrow}$, then apply Lemma 4.11; this would show $f^\infty \overset{s}{\Longrightarrow}$. To that end, we build a strictly increasing sequence of (autonomous) transitions $f^{n\times p} \overset{s_n}{\Longrightarrow} g_n$, such that $g_n(x) \overset{s'n}{\Longrightarrow}$ and such that $s = s_n s'_n$ for all n (s_n and s'_n are traces whose concatenation is s). Here p is such that f^p is autonomous (we always unfold p times at once). Strictly increasing means that the transition $f^{(n+1)\times p} \overset{s_{n+1}}{\Longrightarrow} g_{n+1}$ can be decomposed as $f^{(n+1)\times p} \overset{s_n}{\Longrightarrow} g_n \circ f^p \overset{s''_n}{\Longrightarrow} g_{n+1}$ for some s''_n ($s_{n+1} = s_n s''_n$). This construction will have to stop, otherwise we build a non-innocuous divergence.

Construction of the sequence. Assume p is such that f^p is autonomous; such a p exists, since f is well-behaved.

We initialise with the empty trace from $f^0 = \mathtt{id}$. Indeed, we have $\mathtt{id}(x) \Rightarrow \mathtt{id}(x) \overset{s}{\Longrightarrow}$, and s is indeed the concatenation of itself with the empty trace.

Then, at step n, suppose we have, for instance, $f^{n\times p} \overset{s_n}{\Longrightarrow} g_n$, and $g_n(x) \overset{s'_n}{\Longrightarrow}$.

- If s'_n is the empty trace, we stop. We have in this case $f^n \overset{s}{\Longrightarrow}$, which concludes.
- Otherwise, we have $f^{(n+1)\times p} \overset{s_n}{\Longrightarrow} g_n \circ f$ (by autonomy). The operators are $\subseteq_{tr}$-operators, and by hypothesis $x \subseteq_{tr} f(x)$, therefore $g_n(x) \subseteq_{tr} g_n \circ f^p(x)$. From there, $g_n \circ f^p(x) \overset{s'_n}{\Longrightarrow}$.

  s'_n is not the empty sequence, so there is $y_1, \ldots, y_n$ for $n \geq 1$ such that $g_n \circ f^p(x) \xrightarrow{\mu_1} y_1 \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_n} y_n$, where $s'_n = \mu_1 \ldots \mu_k$. Take $\mu_1 \ldots \mu_i$ the longest prefix of s'_n such that there is operators $h_1, \ldots, h_i$ such that $g_n \circ f^p \xrightarrow{\mu_1} h_1 \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_i} h_i$. It cannot be empty: indeed, f^p is autonomous and composition respects autonomy, hence so is $g_n \circ f^p$, so $g_n \circ f^p(x) \xrightarrow{\mu_1} y_1$ is always autonomous. We take g_{n+1} to be h_i, s'_{n+1} to be the trace corresponding to $\mu_{i+1} \ldots \mu_k$ (i.e., removing all τs) and s_{n+1} to be $s_n s''_n$, where s''_n is the trace corresponding to $\mu_1 \ldots \mu_i$ (again, removing all τs).

  Then, we indeed have $f^{n\times p} \circ f^p = f^{(n+1)\times p} \overset{s_{n+1}}{\Longrightarrow} g_{n+1}$, and $g_{n+1}(x) \overset{s'_{n+1}}{\Longrightarrow}$, where $s = s_{n+1} s'_{n+1}$.

Suppose now that the construction given above never stops. To make the argument clearer, we reason up to $=_f$ (we identify all operators that are $=_f$). We know that $g_n \circ f^p \overset{s''_n}{\Longrightarrow} g_{n+1}$, therefore, by applying Lemma 4.5 as many times as needed (just as in Lemma 4.11), we get that $g_n \circ f^\infty \overset{s''_n}{\Longrightarrow} g_{n+1} \circ f^\infty$.

This gives an infinite sequence of transitions starting from f^∞: $f^\infty \overset{s''_0}{\Longrightarrow} g_1 \circ f^\infty \overset{s''_1}{\Longrightarrow} \ldots$. We observe that in the latter sequence, every step involves at least one transition (the s''_i are supposed non-empty, otherwise we would have stopped). Moreover, the sequence

$s''_1 s''_2 \ldots$, when removing all τs, is a prefix of s. Therefore there is a finite number of visible actions occurring in this infinite sequence. Therefore f^∞ is divergent.

Furthermore, this divergence is not innocuous: at every step, we know that the transition $g_{n+1}(x) \xRightarrow{s''_{n+1}}$ is not autonomous, otherwise s''_{n+1} would be part of s_n. Hence, there cannot be such a divergence $f^m \xRightarrow{s''_0} \xRightarrow{s''_1} \ldots$, as this would imply $g_{n+1}(x) \xRightarrow{s''_{n+1}}$ is autonomous for n such that $n \times p \geq m$. □

In contrast, the theorems fail for *infinitary trace equivalence*, $\approx^\infty_{tr}$ (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 [San15]. As a counterexample, we consider equation $X = a + a.X$, whose syntactic solution has no divergences. The process $P \stackrel{\triangle}{=} \sum_{n>0} a^n$ is a solution, yet it is not $\approx^\infty_{tr}$-equivalent to the syntactic solution of the equation, because the syntactic solution has an infinite trace involving a transitions. This phenomenon is due to the presence of infinitary observables.

Preorders. We show how the theory for equivalences can be transported onto *preorders*. This means moving to *systems of pre-equations*, $\{X_i \leq E_i\}_{i\in I}$. With preorders, our theorems have a different shape: we do not use pre-equations to reason about unique solution – we expect interesting pre-equations 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 pre-equation is below its syntactic solution.

In the LTS we consider, an equation does not always have a solution. We thus extend the original LTS with the transitions corresponding to the autonomous transitions of the syntactic solution f^∞.

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 [vG90].

In the abstract setting, the body of the pre-equations are functions. Then the theorems give us conditions under which, given a pre-equation $X \leq f(X)$ 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 4.9; other theorems are transported in a similar manner, both the statements and the proofs.

Theorem 4.13. *Let $f \in \mathcal{O}$ be a well-behaved operator. If all divergences of f^∞ are innocuous, then, given a preorder $\leq \in \{\subseteq_{tr}, \subseteq_{tr^\infty}, \leq_s\}$, whenever $x \leq f(x)$ we also have $x \leq f^\infty$, for any state x.*

Proof. (1) For $\subseteq_{tr}$, the proof is given as the item 2 of the proof of Theorem 4.12.

(2) For $\subseteq_{tr^\infty}$, the proof is very similar to the previous proof: simply consider infinite traces, and rather that disjunct over whether or not the construction stops (the construction cannot stop), disjunct over whether the full trace is captured, or only τ actions occur from a moment in the construction.

(3) For $\leq_s$ The proof is strictly included in the proofs of Theorems 3.7 or 4.9. □

Theorem 4.13 intuitively says that the syntactic solution of a pre-equation is *maximal* among all solutions. Note that, in contrast with equations, Theorem 4.13 and the theory of pre-equations also work for *infinitary trace inclusion*.

The opposite direction for pre-equations, namely $\{X_i \geq E_i\}_{i\in I}$ is less interesting; it means that the syntactic solution is *minimal* among the solutions. This property is usually

easy to obtain for a behavioural preorder, and we do not need hypotheses such as autonomy, well-behavedness, or non divergence, as stated in the following proposition:

Proposition 4.14. *Let $f \in \mathcal{O}$ be an operator, $\leq \in \{\subseteq_{tr}, \leq_s\}$, and x such that $f(x) \leq x$. Then, $f^\infty \leq x$.*

Proof. (1) For $\subseteq_{tr}$, the proof is given as item 1 of the proof of Theorem 4.12.
(2) For $\leq_s$, the proof is very similar to the above proof for $\subseteq_{tr}$. □

However, Proposition 4.14 may fail for preorders with infinitary observables, such as infinitary trace inclusion. This is why the theory of equations fails for infinitary trace equivalence. Indeed, if P is a solution for the equation $E \subseteq_{tr^\infty} X$ (whether or not E is divergence-free), then $K_E \subseteq_{tr^\infty} P$ does not necessarily hold. The equation $X = a + a.X$, which is discussed after Theorem 4.12, provides a counter-example to illustrate this observation.

Remark 4.15. Suppose that x and y are such that $x \leq f(x)$ and $y \leq x$. Then we do not necessarily have $y \leq f(y)$. Take for instance, in CCS, $P = a.b$, $Q = a \mid b$ and $E = a.X + b.X$. Then $Q \leq_s E[Q]$, and $P \leq_s Q$, however $P \not\leq_s E[P]$.

4.3. **Rule formats.** A way to instantiate the results in Sections 4.1 and 4.2 is to consider *rule formats* [AFV01, MRG07]. 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 terms or processes of the language).

One of the most common formats is GSOS [BIM95, BIM88, vG05, FvG16]. The format guarantees that the constructs of the language preserve autonomy. This allows to show that the set of unary contexts (meaning, contexts that can be filled with one term) in a GSOS language constitute a set of operators (Definition 4.3 (when seen as functions from terms to terms), and allows us to apply Theorem 4.9 to any GSOS language.

We now use x, y for *state variables*, and t, u for terms that are part of the considered GSOS language (i.e., for elements of the set of states). We use c, d for contexts, seen as functions from the set of states to itself. We write $\mathtt{op}$ for a construct of the language (a function symbol).

For a complete overview of rule formats and for the relevant definitions, we refer the reader to [BIM88]. We recall the format of GSOS rules below, slightly modified in order to use contexts:

$$\frac{\{x_i \xrightarrow{\mu_{i,j}} y_{i,j} \mid i \in I,\ 1 \leq j \leq m_i\} \cup \{x_j \xrightarrow{\mu'_{j,k}} \not\mid \ j \in J,\ 1 \leq k \leq n_j\}}{\mathtt{op}(\widetilde{x}) \xrightarrow{\mu} c(\widetilde{x}, \widetilde{y})}$$

I, J are fixed subsets of $[1, n]$, where n is the length of $\widetilde{x}$. For $i \in I$ (resp. $j \in J$), m_i (resp. n_j) is a fixed integer. $\widetilde{y}$ is the list consisting of all $y_{i,j}$ for $i \in I$ and $1 \leq j \leq m_j$, as well as all $y_{j,k}$ for $j \in J$ and $1 \leq k \leq n_j$. c is a context belonging to the language.

Lemma 4.16. *The set of unary contexts $\mathcal{O}$ of a language defined within the GSOS format is such that composition in $\mathcal{O}$ preserves autonomy on $\mathcal{O}$.*

Proof. Let $\mathcal{L}$ be such a language. We reason by induction over the contexts (which are defined inductively using the constructs of $\mathcal{L}$).

- The empty context (i.e., the 'hole', corresponding to the identity function $\mathtt{id}$) preserves autonomy: if $c \in \mathcal{O}$ is autonomous, then so is $\mathtt{id} \circ c = c$.
- Consider a n-ary construct of the language (a function symbol), $\mathtt{op}$, and a context $c = \mathtt{op}(c_1, \ldots, c_n)$, where the c_i's are unary contexts.

  We have to show that so is $c \circ c'$, for c' autonomous. Consider for this a transition $c \circ c'(t) \xrightarrow{\mu} u$ $(t, u \in \mathcal{L})$, we prove that this transition is autonomous. The last rule of the derivation tree of this transition is an instance of a GSOS rule as seen above:

$$\frac{\{x_i \xrightarrow{\mu_{i,j}} y_{i,j} \mid i \in I,\ 1 \leq j \leq m_i\} \cup \{x_j \xrightarrow{\mu'_{j,k}} \not\mid \ j \in J,\ 1 \leq k \leq n_j\}}{\mathtt{op}(\widetilde{x}) \xrightarrow{\mu} c_0(\widetilde{x}, \widetilde{y})}$$

  (I, J and the m_is and n_js being fixed, as well as c_0).

  By definition $c \circ c'(t) = \mathtt{op}(c_1 \circ c'(t), \ldots, c_n \circ c'(t))$. Therefore the x_i must be instantiated by the terms $c_i \circ c'(t)$. Hence there are $u_{i,j} \in \mathcal{L}$ such that $c_i \circ c'(t) \xrightarrow{\mu_{i,j}} u_{i,j}$ for $i \in I,\ 1 \leq j \leq m_i$. As well, $c_j \circ c'(t) \xrightarrow{\mu_{j,k}} \not\mid$ for $j \in J,\ 1 \leq k \leq n_j$. Lastly, writing $u = c_0(\widetilde{c} \circ c'(t), \widetilde{u})$ and $c \circ c'(t) \xrightarrow{\mu} c_0(\widetilde{c} \circ c'(t), \widetilde{u})$.

  By induction hypothesis, each of the c_i's preserves autonomy under composition on $\mathcal{O}$, hence $c_i \circ c'$ is autonomous for all i. Hence there are autonomous transitions $c_i \circ c' \xrightarrow{\mu_{i,j}} d_{i,j}$, where $d_{i,j}$ are contexts such that $u_{i,j} = d_{i,j}(t)$.

  Furthermore, for any $t' \in \mathcal{L}$, $c_j \circ c'(t') \xrightarrow{\mu_{j,k}} \not\mid$: if there was such a t', by autonomy of $c_j \circ c'$, we would have $c_j \circ c'(t) \xrightarrow{\mu_{j,k}} \not\mid$.

  Therefore, for any $t' \in \mathcal{L}$, the premises of the above GSOS rule hold; hence we can deduce $c \circ c'(t') \xrightarrow{\mu} c_0(\widetilde{c} \circ c'(t'), \widetilde{d}(t))$. Thus there is an autonomous transition $c \circ c' \xrightarrow{\mu} c_0(\widetilde{c} \circ c', \widetilde{d})$, and $u = (c_0(\widetilde{c} \circ c', \widetilde{d}))(t)$. The transition $c \circ c'(t) \xrightarrow{\mu} u$ is indeed autonomous. This concludes the proof. □

Proposition 4.17. *Consider the set of unary contexts of a language defined within the GSOS formats, and suppose that bisimilarity is preserved by these contexts. This set constitutes a set of operators.*

Proof. This is a direct consequence of Lemma 4.16: for any language, the empty context acts as the identity function over the set of terms, and by construction contexts can be composed; by hypothesis, we consider contexts that respect bisimilarity. All is left to check is clause 3 of Definition 4.3, i.e., that composition preserves autonomy over $\mathcal{O}$: this is give by Lemma 4.17. □

Some GSOS rule formats guarantee congruence for weak bisimilarity [vG05, FvG16], which gives clause 4 of Definition 4.3. This condition could actually be relaxed, by imposing congruence only for certain forms of contexts, according to how variables occur in the equations. For instance, if certain syntactical constructs are not used, we do not need congruence to hold for these constructs. This can be useful, for instance, to allow non-guarded sums in CCS: in sums, we require variable occurrences to be under prefixes (and the sum itself need not between prefixed processes).

Remark 4.18. Proposition 4.17 makes it possible to apply Theorems 4.7 and 4.9 to any GSOS language that additionally preserves bisimilarity. However the formulation of the 'syntactic solution of an equation' and of 'innocuous divergence' in Section 4.1 is not as easy to grasp as in CCS.

An alternative would be to assume that the language has constants for recursively defined processes, which would allow us to remain closer to the setting of CCS, as exposed in Section 3.

We now state the abstract version of Theorem 3.7. For this, we rely on the definition of divergence and innocuous divergence from Section 4.1, and use contexts of a GSOS language to stand for operators.

Theorem 4.19. *Consider a language whose constructs have SOS rules in the GSOS format and preserve $\approx$. Consider an equation of the form $X = E$, where E is a function corresponding to a context of the language (seen as an equation expression), that is autonomous and has only innocuous divergences,*

Then either the equation has no solution or it has a unique solution for $\approx$.

Proof. This is a direct consequence of Theorem 4.9 and Proposition 4.17. Consider the context C such that $C[X]$ and E are syntactically equal. Then $C \in \mathcal{O}$, where $\mathcal{O}$ is the set of unary contexts of the language; by Proposition 4.17, it is a set of operators. Using Theorem 4.9 we deduce that the equation has a unique solution. □

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. Definitions of guardedness for SOS specifications from [Vaa92] can also be applied here.

tyft/tyxt formats. We briefly discuss other widely studied formats, the tyft/tyxt formats [GV92]. In those formats, 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 4.3), hence Theorem 4.9 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:

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

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 $\approx$.

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, $A\pi$ [SW01] (addressing the full π-calculus would also be possible, but somewhat more involved). The theory of $A\pi$ is simpler than that of the synchronous π-calculus. Notably, bisimulation does not require closure under name instantiation.

Equation expressions, processes and systems of equations. For the π-calculus, we inherit the notations from CCS. $a, b, \ldots, x, y, \ldots$ range over the infinite set of names, and $X, Y, \ldots$ range over equation variables.

In the π-calculus, the solutions of equations are *process abstractions*, i.e., processes which are made parametric over their free names. This allows us to work with closed agents, which makes the treatment of equations easier (in particular w.r.t. name capture).

In order to define the syntactical entities involved in the reasoning about the π-calculus, it is useful to start by introducing equation expressions, ranged over using $E, E', \ldots$, as follows:

$$F \;:=\; K \;\big|\; X \qquad \text{(abstraction identifiers)}$$

$$E \;:=\; \mathbf{0} \;\big|\; a(\widetilde{b}).\, E \;\big|\; \overline{a}\langle\widetilde{b}\rangle \;\big|\; E_1 \mid E_2 \;\big|\; \boldsymbol{\nu} a\, E \;\big|\; !a(\widetilde{b}).\, E \;\big|\; F\langle\widetilde{a}\rangle \qquad \text{(equation expressions)}$$

Inputs of the form $c(\widetilde{a}).\, E$ and $!c(\widetilde{a}).\, E$, and restrictions $\boldsymbol{\nu} a\, E$ are binders for the names $\widetilde{a}$ and a, with scope E.

Replication could be avoided in the syntax since it can be encoded with recursion. However replication 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 (we only allow replicated inputs, which are guarded).

The grammar for equation expressions includes the application construct $F\langle\widetilde{a}\rangle$, used to instantiate the formal parameters of the abstraction identifier F with the actual parameters $\widetilde{a}$. Indeed, since the operational semantics of the π-calculus makes use of name substitution, the body of recursive definitions and of equations is parametrised over a set of names (as will be apparent below).

Processes, ranged over using $P, Q, \ldots$, are defined as equation expressions in which no occurrence of equation variables is allowed (therefore the grammar for processes, P, is like the grammar for E above, except that K replaces F).

Process abstractions are of the form $(\widetilde{a})\, P$, where $\widetilde{a}$ is a non-empty tuple of distinct names. In $(\widetilde{a})\, P$, names $\widetilde{a}$ are bound in P, giving rise to the usual notion of bound and free names. A process abstraction is *closed* if it has no free name.

Agents. Processes, process abstractions, and constant identifiers form the set of *agents*:

$$A \quad := \quad P \;\big|\; (\widetilde{a})\, P \;\big|\; K \qquad \text{(agents)}$$

An agent is *closed* if it does not have free names.

Recursive definitions. K is used to stand for recursively defined abstractions. Each K should have a defining equation of the form $K \stackrel{\triangle}{=} (\widetilde{a})\, P$, with the additional constraint that $(\widetilde{a})\, P$ is closed (i.e., it has no free name).

Systems of equations. In order to write equations, we need to work with *equation abstractions*, written $(\widetilde{a})\, E$, where $\widetilde{a}$ is a non-empty tuple of distinct names. Again, in $(\widetilde{a})\, E$, names $\widetilde{a}$ are bound in E, giving rise to the usual notion of bound and free names. We say that $(\widetilde{a})\, E$ is *closed* if it has no free name.

An equation is of the form $X = (\widetilde{a})\, E$, where the body $(\widetilde{a})\, E$ of the equation is closed, and can contain the equation variable X. A solution of such an equation is given by a *closed* process abstraction of the form $(\widetilde{b})\, P$ (or, in the case where $\widetilde{a}$ is empty, by a closed process). Imposing this condition allows us to ignore difficulties related to name capture (if, for instance, we substitute (non closed) $P = a(x).\, Q$ for X in the equation expression $E = \boldsymbol{\nu} a\; X$, name a gets captured by the restriction).

Equation variables occurring in equation expressions can be instantiated with process abstractions. When doing so, we instantiate names in the process abstractions according to the usages of the equation variable in the equation expression; this operation is done implicitly, and does not correspond to a computation step. We illustrate this on an example:

Example 5.1. Consider the equation expression $E = \overline{c}\langle d\rangle \mid X\langle\widetilde{a}\rangle$, having X as only equation variable. Then $E[(\widetilde{b})\,P]$ stands for $\overline{c}\langle d\rangle \mid P\{\widetilde{a}/\widetilde{b}\}$. Note that the equation associated to equation expression E is of the form $X = (c, d, \widetilde{a})\,E$.

We can also instantiate equation variables with equation abstractions: if E_1 is an equation expression and $(\widetilde{a})\,E_2$ is an equation abstraction, then $E_1[E_2]$ stands for the equation expression obtained by instantiating the variable in E_1 with $(\widetilde{a})\,E_2$.

The definitions above for equations are extended to systems of equations; in particular, the solution of a system of equations is given by a tuple of process abstractions.

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

Sorting. Since the calculus is polyadic, we assume a *sorting system* [Mil99] 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 equation variables, it is intended that each variable has a sort so that we know the number and the sorts of their parameters. An equation expression is also sorted, and, accordingly, composition of equation expressions is supposed to be well-sorted.

We will not present the sorting system because it is well-studied (cf. [Mil99, SW01]), and adapting it to our setting raises no particular difficulty. 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 $\overline{a}\langle\widetilde{b}\rangle$. Free and bound names of an action are defined as usual. Figure 5 presents the transition rules for $A\pi$. We maintain from CCS the notations for transitions, such as $\Rightarrow$, $\stackrel{\hat{\mu}}{\Longrightarrow}$, and so on.

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

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

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

Figure 5: $A\pi$: Labelled Transition Semantics

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 [SW01]. Similarly, the extension of bisimilarity to agents only considers fresh names: $A \approx A'$ if $A\langle\widetilde{a}\rangle \approx A'\langle\widetilde{a}\rangle$ where $\widetilde{a}$ is a tuple of fresh names (as usual, of the appropriate sort).

Example 5.2. If K is the syntactic solution of the equation $X = (a)\ !a(x).\, X\langle x\rangle$, then we have $K\langle b\rangle \xrightarrow{b(y)} K\langle y\rangle \mid K\langle b\rangle$.

Theorems 3.5 and 3.7 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 5.3 (Unique solution in Aπ)**.** *A guarded system of equations whose syntactic solutions do not contain divergences has a unique solution for $\approx$.*

Proof. Because we only consider closed abstractions, all names are instantiated when performing a substitution of a variable by an abstraction. This allows us to use the same proof as for Theorem 3.7 in CCS. □

Theorem 5.4 (Unique solution with innocuous divergences in Aπ)**.** *A guarded system of equations whose syntactic solutions only have innocuous divergences has a unique solution for $\approx$.*

Proof. The proof of Theorem 5.3 has to be modified exactly as in the CCS case, where the proof of Theorem 3.5 is modified to establish Theorem 3.7. □

As in CCS, the guardedness condition can be removed if the rule for the unfolding of a constant produces a τ-transition.

We now state an analogue of Lemma 3.10 for $A\pi$, giving a sufficient condition to guarantee that a system of equations only has innocuous divergences. This condition is decidable, and sufficient for the application in Section 5.2. We leave the study of finer conditions for future work.

Lemma 5.5. *Consider a well-sorted system of equations $\widetilde{X} = \widetilde{E}$ in $A\pi$ (in particular, for each i, the sort of X_i and of E_i are the same).*

Suppose that there is a sort of names such that names of that sort are never used in subject output position. If for each i there is n_i such that in $(E_i)^{n_i}$, each equation variable occurs underneath an (input) prefix of that sort, then the system has only innocuous divergences.

In earlier sections about CCS, we pointed out the connection between techniques based on unique solution of equations and 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.

The full π-calculus. In the full π-calculus, including the output prefix $\overline{a}\langle\widetilde{b}\rangle.\,P$, ground bisimilarity is not a congruence. One has therefore to use other forms of bisimilarity. The most used is *early bisimilarity*; correspondingly one uses an early transition system, where the parameters of inputs are instantiated with arbitrary (i.e., not necessarily fresh) names, as in $a(\widetilde{x}).\,P \xrightarrow{a\langle\widetilde{b}\rangle} P\{\widetilde{b}/\widetilde{x}\}$. Modulo the move to the early setting, the theory exposed for the asynchronous π-calculus also holds in the full π-calculus.

However, when considering a different LTS, such as the early LTS, divergences that did not exist in the ground LTS may arise, and as such, prevent Theorems 5.3 or 5.4 to be used. It is even possible that, by considering a different LTS, an equation that had a unique solution loses this property. We leave the study of this question for future work.

5.2. **An application: encoding of the call-by-name λ-calculus.** To show an extended application of our techniques for the π-calculus, we revisit the proof of full abstraction for Milner's encoding of the call-by-name (or lazy) λ-calculus into $A\pi$ [Mil92] 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.

We assume the standard concepts of free and bound variables and substitutions, and identify α-convertible terms. The terms in Λ are sometimes call *open* to distinguish them from the subset of *closed* terms – those without free variables. The rules defining the call-by-name strategy, defined on open terms, are the following:

$$(\lambda x.\,M)N \to M\{N/x\} \qquad\qquad \frac{M \to M'}{M\ N \to M'\ N}$$

As usual $\Rightarrow$ is the reflexive and transitive closure of the single-step reduction $\to$.

We write $M \Uparrow$ if M diverges. If M is an open λ-term, then either M diverges, or $M \Rightarrow \lambda x.\,M'$ (for some x and M'), or $M \Rightarrow x\ M_1\ \ldots\ M_n$ (for some x, M_1, $\ldots$, M_n).

The following notion of tree is used to provide a semantics for the call-by-name λ-calculus.

Definition 5.6 (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 $M \Rightarrow \lambda x.\,M'$, then $LT(M)$ is the tree with a root labelled with "$\lambda x.$", and $LT(M')$ as its unique descendant.
(3) If $M \Rightarrow x\ M_1\ \ldots\ M_n$, 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 (whereby 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 5.7. 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.

Theorem 5.8 [San00, SX18]**.** *Let M and N be two λ-terms; then $LT(M) = LT(N)$ iff $M \simeq^o N$.*

Milner's encoding of the call-by-name λ-calculus into $A\pi$ [Mil92] is defined as follows:

$$[\![\lambda x.\, M]\!] \stackrel{\triangle}{=} (p)\ p(x,q).\ [\![M]\!]\langle q\rangle \qquad\qquad [\![x]\!] \stackrel{\triangle}{=} (p)\ \overline{x}\langle p\rangle$$

$$[\![M\ N]\!] \stackrel{\triangle}{=} (p)\ \boldsymbol{\nu} r, x\ ([\![M]\!]\langle r\rangle \mid \overline{r}\langle x, p\rangle \mid\ !x(q).\, [\![N]\!]\langle q\rangle)$$

Function application is translated into a particular form of parallel combination of two agents, the function and its argument; β-reduction is then modeled as process interaction. Since the syntax of the π-calculus only allows for the transmission of names along channels, the communication of a term is simulated by the communication of a *trigger* for it. The translation of a λ-term is an abstraction that is parametric on a name, the *location* of the λ-term, which is intuitively the name along which the term, as a function, will receive its argument. Precisely, the encoding of a term receives two names along its location p: the first is a trigger for its argument and the second is the location to be used for the next interaction.

The full abstraction theorem for the encoding [San00, SX18] 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 [San00, SX18] first establishes an 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 amounts to following the defining clauses of open bisimulation (Definition 5.7). In contrast, completeness involves enhancements of the bisimulation proof method, notably 'bisimulation up to context and expansion'. In the latter, *expansion* is an auxiliary preorder relation, finer than weak bisimilarity. 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 (expansion appears within the statements of operational correspondence).

Below we show that, by appealing to unique solution of equations, completeness can be proved by defining an appropriate system of equations, each equation 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 5.9), 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 (Lemma 5.10); (*iii*) a Lemma for rearranging parallel composition and restrictions in the process encoding $x\ M_1\ \ldots\ M_n$ (Lemma 5.11).

Lemma 5.9 (Validity of β-reduction, [San00])**.** *For $M \in \Lambda$, if $M \to M'$ then $[\![M]\!] \approx [\![M']\!]$.*

Lemma 5.10. *If M diverges, then $[\![M]\!] \approx (p)\,\mathbf{0}$.*

Lemma 5.11 [San00]**.** *For any $M_1, \ldots, M_n \in \Lambda$, and variable x, we have*

$$[\![x\ M_1\ \ldots\ M_n]\!] = (p)\,(\boldsymbol{\nu} r_0, \ldots, r_n\,)\big(\overline{x}\langle r_0\rangle \mid \overline{r_0}\langle r_1, x_1\rangle \mid \ldots \mid \overline{r_{n-1}}\langle r_n, x_n\rangle \mid\ !x_1(q_1).\, [\![M_1]\!]\langle q_1\rangle \mid \ldots \mid\ !x_n(q_n).\, [\![M_n]\!]\langle q_n\rangle\big)\ .$$

Theorem 5.12 (Completeness, [San00]). *For $M, N \in \Lambda$, $LT(M) = LT(N)$ implies $[\![M]\!] \approx [\![N]\!]$.*

Proof. Suppose M_0 and N_0 are two λ-terms with the same LT. We define a system of equations, whose solutions are obtained from the encodings of M_0 and N_0. We will then use Theorem 5.4 to deduce $[\![M_0]\!] \approx [\![N_0]\!]$.

Since M_0 and N_0 have the same LT, then by Theorem 5.8 there is an open bisimulation $\mathcal{R}$ containing the pair (M_0, N_0). 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 consider a pair (M, N) in $\mathcal{R}$, and explain how the corresponding equation is defined. We assume an ordering of the λ-calculus variables so to be able to view a finite set of variables as a tuple. This allows us to write $\widetilde{x}$ for the variables appearing free in M or N.

The equations are the translation of the clauses of Definition 5.7, assuming a generalisation of the encoding to equation variables by adding the clause: $[\![X_{M,N}]\!] \stackrel{\text{def}}{=} (\widetilde{x}, p) X_{M,N} \langle \widetilde{x}, p \rangle$.

Since $M \mathcal{R} N$, then either both M and N diverge, or they satisfy one of the two clauses of Definition 5.7.

- If M, N are both divergent, then the equation is $X_{M,N} = (\widetilde{x}, p)!\tau$. (as $!\tau \approx [\![\Omega]\!]$)
- If M, N satisfy clause 1 of Definition 5.7, the equation is

$$X_{M,N} = (\widetilde{x}, p)\, [\![\lambda x.\, X_{M',N'}]\!] \langle p \rangle \ , \quad \text{that is,} \quad X_{M,N} = (\widetilde{x}, p)\, p(x,q).\, X_{M',N'} \langle \widetilde{y}, q \rangle \ ,$$

  where $\widetilde{y}$ are the free variables in M', N'.
- If M, N satisfy clause 2 of Definition 5.7, 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 (Lemma 5.11), can be written

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

  where $\widetilde{x_i}$ are the free variables in M_i, N_i.

Essentially, each equation above represents the translation of a specific node of the LT for M and N.

We then rely on Lemma 5.5 to show that the equations may only produce innocuous divergences. It is easy to check that the syntactic condition holds: a location name may only appear once (in input position); a trigger name either appears once (as a replicated input), or it only appears in output position.

Now, for $(M, N) \in \mathcal{R}$, define $F_{M,N}$ as the abstraction $(\widetilde{x}, p)[\![M]\!]\langle p \rangle$, and similarly $G_{M,N} \stackrel{\triangle}{=} (\widetilde{x}, p)[\![N]\!]\langle p \rangle$. Lemma 5.9 allows us to show that the set of all such abstractions $F_{M,N}$ yields a solution for the system of equations, and similarly for $G_{M,N}$.

We reason by cases, following Definition 5.7.

- If clause (1) of Definition 5.7 holds, then $M \Rightarrow x\ M_1\ \ldots\ M_n$ and we have

$$\begin{aligned} F_{M,N} &= (\widetilde{y}, p)[\![M]\!]\langle p \rangle \\ &\approx (\widetilde{y}, p)[\![x M_1 \ldots M_n]\!]\langle p \rangle && \text{(by Lemma 5.9)} \\ &= (\widetilde{y}, p) p(x,q).\, [\![M']\!] q \\ &= (\widetilde{y}, p)\big(p(x,q).\, X_{M',N'} \langle \widetilde{z} \rangle\big)\{F_{M',N'}/X_{M',N'}\} \end{aligned}$$

  where $\widetilde{z}$ is a subset of $x, \widetilde{y}$, containing the free variables of M and N.

- If clause (2) holds, then $M \Rightarrow \lambda x.\, M'$ and we have

$$\begin{aligned} F_{M,N} &= (\widetilde{y}, p)[\![M]\!]\langle p\rangle \\ &\approx (\widetilde{y}, p)(\boldsymbol{\nu} r_0, \ldots, r_n\,)\big(\overline{x}\langle r_0\rangle \mid \overline{r_0}\langle r_1, x_1\rangle \mid \ldots \mid \overline{r_{n-1}}\langle r_n, x_n\rangle \mid \\ &\qquad !x_1(q_1).\,[\![M_1]\!]q_1 \mid \ldots \mid !x_n(q_n).\,[\![M_n]\!]q_n\,\big) \qquad \text{(by Lemmas 5.9 and 5.11)} \\ &\approx (\widetilde{y}, p)(\boldsymbol{\nu} r_0, \ldots, r_n\,)\big(\overline{x}\langle r_0\rangle \mid \overline{r_0}\langle r_1, x_1\rangle \mid \ldots \mid \overline{r_{n-1}}\langle r_n, x_n\rangle \mid \\ &\qquad !x_1(q_1).\,X_{M_1,N_1}\langle \widetilde{x_1}, q_1\rangle \mid \ldots \mid !x_n(q_n).\,X_{M_n,N_n}\langle \widetilde{x_n}, q_n\rangle\,\big) \\ &\qquad \{F_{M_1,N_1}/X_{M_1,N_1}, \ldots, F_{M_n,N_n}/X_{M_n,N_n}\} \end{aligned}$$

- If neither clause (1) or (2) hold, meaning both M and N diverge, then we have

$$\begin{aligned} F_{M,N} &= (\widetilde{x}, p)[\![M]\!]\langle p\rangle \\ &\approx (\widetilde{x}, p)\mathbf{0} \qquad \text{(by Lemma 5.10)} \\ &\approx (\widetilde{x}, p)!\tau \end{aligned}$$

Since $F_{M,N}$ and $G_{M,N}$ are both solutions of the system of equations, they are bisimilar, which finally allows us to deduce that $[\![M_0]\!] \approx [\![N_0]\!]$ □

6. Conclusions & 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. The technique by Pous, in particular, is arguably more complex, both in its definition and its application, than the unique solution theorems presented here.

As said above, the comparison with up-to techniques could also help understanding the need for the closure under substitutions in up to context techniques for name-passing calculi such as the asynchronous π-calculus. Surprisingly, our unique solution techniques, despite the strong similarities with up-to context techniques, do not require the closure under substitutions. However, currently it is unclear how to formally relate bisimulation enhancements and 'unique solution of equations' in name-passing calculi.

Up-to techniques have been analysed in an abstract setting using lattice theory [Pou16] and category theory [BPPR17, RBR13]. It could be interesting to do the same for the unique-solution techniques, to study their connections with up-to techniques, and to understand which equivalences can be handled (possibly using, or refining, the abstract formulation presented in Section 4).

In comparison with the enhancements of the bisimulation proof method, the main drawback of the techniques exposed in this paper 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 3.10). Various techniques for checking divergence in concurrent calculi exist in the literature, including type-based techniques [YBH04, DHS10, DS06]. 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.

Several studies in functional programming and type theory rely on type-based methods to insure that coinductive definitions are productive, i.e., do not give rise to partially defined functions (in order to preserve logical consistency) [Nak00, AM13]. Understanding whether these approaches can be adapted to analyse divergences and innocuous divergences in systems of equations is a topic for future investigations.

Acknowledgements. This work was supported by Labex MILYON/ANR-10-LABX-0070, by the European Research Council (ERC) under the Horizon 2020 programme (CoVeCe, grant agreement No 678157), by H2020-MSCA-RISE project 'Behapi' (ID 778233), and by the project ANR-16-CE25-0011 REPAS.

References

[AFV01] Luca Aceto, Wan Fokkink, and Chris Verhoef. *Handbook of Process Algebra (J.A. Bergstra and A. Ponse and S.A. Smolka, editors)*, chapter Structural operational semantics. Elsevier Science, 2001.

[AM13] Robert Atkey and Conor McBride. Productive coprogramming with guarded recursion. In Greg Morrisett and Tarmo Uustalu, editors, *ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA - September 25 - 27, 2013*, pages 197–208. ACM, 2013.

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

[BHR84] Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[BIM88] 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.

[BIM95] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.

[BPPR17] Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. A general account of coinduction up-to. *Acta Inf.*, 54(2):127–190, 2017.

[BR84] 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.

[BW90] J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge University Press, New York, NY, USA, 1990.

[DHS10] Romain Demangeon, Daniel Hirschkoff, and Davide Sangiorgi. Termination in impure concurrent languages. In *CONCUR 2010 - Concurrency Theory, 21th International Conference*, pages 328–342, 2010.

[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.

[DHS18] Adrien Durier, Daniel Hirschkoff, and Davide Sangiorgi. Eager functions as processes. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 364–373. ACM, 2018.

[DS06] Yuxin Deng and Davide Sangiorgi. Ensuring termination by typability. *Inf. Comput.*, 204(7):1045–1082, 2006.

[Dur17] Adrien Durier. Divergence and unique solution of equations in an abstract setting, Coq formal proof. Available at `https://github.com/adurier/uniquesolution/`, 2017.

[FvG16] 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.

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

[GV92] Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inf. Comput.*, 100(2):202–260, 1992.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

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

[Mil92] Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[Mil99] Robin Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999.

[MRG07] 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.

[Nak00] Hiroshi Nakano. A modality for recursion. In *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*, pages 255–266. IEEE Computer Society, 2000.

[Pou08] Damien Pous. *Up to techniques for bisimulations*. PhD thesis, ENS Lyon, February 2008.

[Pou16] 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.

[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.

[RBR13] 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.

[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.

[San00] Davide 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.

[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*, pages 421–432, 2015.

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

[SX18] Davide Sangiorgi and Xian Xu. Trees from functions as processes. *Logical Methods in Computer Science*, 14(3), 2018.

[Vaa92] Frits W. Vaandrager. Expressiveness results for process algebras. In *Sematics: Foundations and Applications, REX Workshop, Beekbergen, The Netherlands, June 1-4, 1992, Proceedings*, pages 609–638, 1992.

[vG90] 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.

[vG05] 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.

[YBH04] Nobuko Yoshida, Martin Berger, and Kohei Honda. Strong Normalisation in the Pi-Calculus. *Information and Computation*, 191(2):145–202, 2004.