Proof Techniques for Behavioural Relations Based on Unique-Solutions of Equations and Inequations

Proof techniques for behavioural relations based on unique-solutions of equations and inequations

Adrien Durier[1], Daniel Hirschkoff[2], and Davide Sangiorgi[3]

[1] Université Paris-Saclay, France,
[2] Univ. Lyon, ENS de Lyon, UCBL, CNRS, LIP, France,
[3] INRIA/Università di Bologna, Italy

Abstract. We give an overview of recent work aiming at strengthening Milner's technique of unique-solution of equations. This technique, originally introduced for CCS, is used to establish weak bisimilarity of processes. We review two improvements of the technique. The first one builds on the contraction behavioural preorder. The second one relaxes the conditions under which unique-solution can be applied by focusing on divergence in processes. We present applications of these techniques to reason about higher-order languages. We discuss how the two approaches, involving contractions and divergences, have been adapted and enriched, and compare them to each other.

1 Introduction

Bisimilarity is employed to define behavioural equivalences and reason about them. In this paper, behavioural equivalences, hence also bisimilarity, are meant to be *weak* because they abstract from internal moves of terms, as opposed to the *strong* ones, 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.

A prominent proof method for bisimulation, put forward by Robin Milner and widely used in his landmark CCS book [22], is the *unique solution of equations*, whereby two tuples of processes are componentwise bisimilar if they are solutions of the same system of equations. This method is important in verification techniques and tools based on algebraic reasoning [2, 30, 31, 10]. Not all equations have a unique solution: for instance any process trivially satisfies $X = X$. In Milner's theorem [22], uniqueness of solutions is subject to limitations: the equations must be 'strongly guarded and sequential', that is, the variables of the equations may only be used underneath a visible prefix and preceded, in the syntax tree, only by the sum and prefix operators. This limits the expressiveness of the technique (since occurrences of other operators above the variables, such as parallel composition and restriction, in general cannot be removed), and its transport onto other languages (e.g., languages for distributed systems or higher-order languages usually do not include the sum operator, which makes the theorem essentially useless). A

comparable technique, involving similar limitations, has been proposed by Hoare in his CSP book [11].

In this paper we summarise some recent works aiming at improving Milner's unique-solution technique. Two main approaches are described. The first consists in replacing equations with special inequations called *contractions* [39]. The second consists in replacing the most severe syntactic constraint in Milner's theorem with a semantic condition that has to do with *divergence* [4, 5].

Other motivations for the work on unique solutions are the following. First, the unique-solution techniques convey the flavour of 'bisimilarity up-to context' techniques, as the equations (or contractions) intuitively bring out those contexts that would be erased in an 'up-to context' proof (indeed there are completeness results with respect to certain forms of up-to context, in CCS-like languages [39]). Thus the study of the unique-solution techniques may bring light into the up-to-context techniques. For instance, in higher-order languages, while there are well-developed techniques for proving congruence [26], up-to context is still poorly understood [15, 13, 12, 41, 25].

Another possible interest for the unique-solution techniques is that they can be transported onto other equivalences, including contextually-defined equivalences such as barbed congruence, and non-coinductive equivalences such as contextual equivalence (i.e., may testing) and trace equivalence.

We present Milner's classic technique of unique-solution of equations for CCS in Section 2. We then introduce the two improvements that have been recently proposed, namely contractions (Section 3) and divergences (Section 4). In Section 5, we discuss extensions and adaptations of these improved techniques. Section 6 presents a formalisation in the HOL4 theorem prover of the work on contractions. We show the expressiveness of the techniques by presenting applications to higher-order languages in Section 7, and we finally compare the approaches involving contractions and involving divergences in Section 8.

2 Background

2.1 CCS

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

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

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

Some standard notations for transitions: $\Longrightarrow$ is the reflexive and transitive closure of $\stackrel{\tau}{\longrightarrow}$, and $\stackrel{\mu}{\Longrightarrow}$ is $\Longrightarrow\stackrel{\mu}{\longrightarrow}\Longrightarrow$ (the composition of the three relations).

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

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

Fig. 1. The LTS for CCS

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

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

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

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

2.2 Equations and Milner's theorem

We recall equations and Milner's theorem, in the setting of CCS. Uniqueness of solutions of equations [22] intuitively says that if a context C obeys certain conditions, then all processes P that satisfy the equation $P \approx C[P]$ are bisimilar with each other.

We use capital letters X, Y, Z for variable of equations. The body of an equation is a CCS expression possibly containing variables.

Definition 2. *Given, for each i of a countable indexing set I, variables X_i, and expressions E_i possibly containing such variables, $\{X_i = E_i\}_{i\in I}$ is a* system of equations. *(There is one equation for each variable X_i.)*

We write $E[\widetilde{P}]$ for the expression resulting from E by replacing each variable X_i with the process P_i

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

- *$\widetilde{P}$ 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* $\widetilde{P} \approx \widetilde{Q}$*. The latter means that for each* i*,* $P_i \approx Q_i$*.*

Definition 4. *A system of equations* $\{X_i = E_i\}_{i \in I}$ *is*

– guarded *if, in each* E_i*, each occurrence of an equation variable is underneath a visible prefix (i.e., a prefix that is not* τ*);*
– sequential *if, in each* E_i*, each occurrence of an equation variable only appears underneath prefixes and sums.*

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 guarded; using ℓ for a visible prefix, $X = \ell.X \mid P$ is guarded but not sequential, whereas $X = \ell.X + \tau.\boldsymbol{\nu} a\,(a.\overline{b} \mid a.\mathbf{0})$, as well as $X = \tau.(a.X + \tau.b.X + \tau)$ are both guarded and sequential.

Theorem 1 (unique solution of equations, [22]). *A system of guarded and sequential equations has a unique solution for* $\approx$*.* □

3 Contractions

A way of weakening the constraints on the unique-solution Theorem 1 is to move from equations to certain inequations called *contractions* [39].

Intuitively, for a behavioural equivalence $\asymp$, its contraction $\succeq_{\asymp}$ is a preorder in which $P \succeq_{\asymp} Q$ holds if $P \asymp Q$ and, in addition, Q has the *possibility* of being at least as efficient as P. That is, if P can do some work (i.e., some interactions with its environment), then Q should be able to do the same work at least as quickly as P (i.e., performing no more τ-steps than those performed by P). Process Q, however, may be nondeterministic, and may have other ways of doing the same work, and these could be slow (i.e., involving more τ-steps than those performed by P). We now apply the idea of contraction to the concrete case of weak bisimilarity.

Definition 5 (bisimulation contraction). *A process relation* $\mathcal{R}$ *is a* bisimulation contraction *if, whenever* $P \mathcal{R} Q$*,*

1. $P \xrightarrow{\mu} P'$ *implies there is* Q' *such that* $Q \xrightarrow{\widehat{\mu}} Q'$ *and* $P' \mathcal{R} Q'$*;*
2. $Q \xrightarrow{\mu} Q'$ *implies there is* P' *such that* $P \xRightarrow{\widehat{\mu}} P'$ *and* $P' \approx Q'$*.*

Bisimilarity contraction*, written* $\succeq_{\mathrm{bis}}$*, is the union of all bisimulation contractions.*

In the first clause Q is required to match P's challenge transition with at most one transition. This makes sure that Q is capable of mimicking P's work at least as efficiently as P. In contrast, the second clause of Definition 5, on the challenges from Q, entirely ignores efficiency: it is the same clause as in weak bisimulation — the final derivatives are even required to be related by $\approx$, rather than by $\mathcal{R}$. (The bisimilarity contraction is similar but coarser than the *expansion relation* [36].)

Theorem 2. $\succeq_{\text{bis}}$ *is a precongruence in CCS.*

A *system of contractions* is defined as a system of equations, except that the contraction symbol $\succeq$ is used in the place of the equality symbol $=$. Thus a system of contractions is a set $\{X_i \succeq E_i\}_{i\in I}$ where I is an indexing set and expressions E_i may contain the contraction variables $\{X_i\}_{i\in I}$.

Definition 6. *Given a behavioural equivalence $\asymp$ and its contraction $\succeq_{\asymp}$, and a system of contractions $\{X_i \succeq E_i\}_{i\in I}$, we say that:*

- *$\widetilde{P}$ is a* solution for $\succeq_{\asymp}$ of the system of contractions *if $\widetilde{P} \succeq_{\asymp} \widetilde{E}[\widetilde{P}]$;*
- *the system has* a unique solution for $\asymp$ *if whenever $\widetilde{P}$ and $\widetilde{Q}$ are both solutions for $\succeq_{\asymp}$ then $\widetilde{P} \asymp \widetilde{Q}$.*

When we reason about bisimilarity, the contraction symbol $\succeq$ is interpreted as the bisimilarity contraction $\succeq_{\text{bis}}$, and the equivalence $\asymp$ as the bisimilarity $\approx$. Thus $\widetilde{P}$ being a solution for $\succeq_{\text{bis}}$ of the system of contractions $\{X_i \succeq E_i\}_{i\in I}$ means that $\widetilde{P} \succeq_{\text{bis}} \widetilde{E}[\widetilde{P}]$; and the system having a unique solution for $\approx$ means that whenever $\widetilde{P}$ and $\widetilde{Q}$ are both solutions for $\succeq_{\text{bis}}$ then $\widetilde{P} \approx \widetilde{Q}$.

Lemma 1. *If a system of equations $\{X_i = E_i\}_{i\in I}$ has a unique solution for $\approx$, then also the corresponding system of contractions $\{X_i \succeq E_i\}_{i\in I}$ has a unique solution for $\approx$.*

The converse of the lemma, in contrast, is false: systems of contractions more easily have a unique solution. Indeed, for contraction, the following weak-guardedness condition is sufficient to have a unique solution.

Definition 7. *A system of contractions $\{X_i \succeq E_i\}_{i\in I}$ is* weakly guarded *if, in each E_i, each occurrence of a contraction variable is underneath a prefix.*

Theorem 3 (unique solution of contractions for $\approx$). *A system of weakly-guarded contractions has a unique solution for $\approx$.*

Example 1. The following contractions have a unique solution for $\approx$:

1. $X \succeq \tau.X$
2. $X \succeq a.\boldsymbol{\nu} a\,(\overline{a} \mid X))$

The corresponding equations do not have a unique solution. The solutions of the contraction (1) are all inactive processes, where a process is *inactive* if it cannot perform visible actions (i.e., if P is the process, then there is no P' and visible action ℓ such that $P \Longrightarrow P' \xrightarrow{\ell}$). The contraction has a unique solution because all inactive processes are bisimilar. Example of solutions for (2) are $a.P$ and $\tau.a.P$, where P is inactive. Any solution of (2) is bisimilar to $a.\mathbf{0}$. □

4 Divergence

In this section we highlight a different approach, whereby one goes back to equations, but adds a condition based in divergence [6].

4.1 Plain divergences

In its basic form, the method (inspired by results by Roscoe in CSP [30, 29]) essentially says that a guarded equation (or system of equations) whose infinite unfolding never produces a divergence has the unique-solution property. We discuss the approach in CCS, as before.

Definition 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 $P = P_0 \xrightarrow{\mu_0} P_1 \xrightarrow{\mu_1} P_2 \xrightarrow{\mu_2} \dots$ 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. 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.

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 given by $K_E \stackrel{\triangle}{=} E[K_E]$. 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), and the syntactic solutions are defined to be the set of mutually recursive constants $\{K_{\widetilde{E},i} \stackrel{\triangle}{=} E_i[\widetilde{K}_{\widetilde{E}}]\}_i$.

Theorem 4 (Unique solution). *A guarded system of equations whose syntactic solutions do not diverge has a unique solution for $\approx$.*

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

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

This procedure necessarily stops: otherwise, we could build an infinite sequence of transitions involving only the unfoldings of E, and with at most one visible transition: this would yield a divergence in the syntactic solution of E. Details may be found in [6].

4.2 Innocuous Divergences

Theorem 4 can be strengthened 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. Such divergences are called *innocuous*. Formally, these divergences are derived by applying only a finite number of times rule `const` of the LTS (see Figure 1) to the constant that represents the syntactic solution of the equation.

Definition 9 (Innocuous divergence). *Consider a guarded system of equations $\widetilde{X} = \widetilde{E}$ and its syntactic solutions $\widetilde{K}_{\widetilde{E}}$. Given some i, a divergence of $K_{\widetilde{E},i}$ is called* innocuous *if, when summing up all usages of rule* `const` *involving one of the $K_{\widetilde{E},j}$s (including the case $j = i$)* in all derivation proofs of the transitions belonging to the divergence*, we obtain a finite number.*

Theorem 5 (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 $\approx$.*

Remark 1. The conditions for unique solution in Theorems 4 and 5 are a mixture of syntactic (guardedness) and semantic (divergence-free) conditions. A purely semantic condition can be used if rule `const` of Figure 1 is modified so that the unfolding of a constant yields a τ-transition:

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

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

5 Extensions

5.1 Other behavioural equivalences

The techniques reviewed in the previous sections can be applied also to other behavioural equivalences, including behavioural equivalences defined on top of inductive observables. As an example, we briefly discuss below the case of *trace equivalence*.

We write $P \stackrel{\mu}{\Longrightarrow}_n P'$ if $P \stackrel{\mu}{\Longrightarrow} P'$ is derived using n strong transitions (i.e., we have $P(\stackrel{\tau}{\longrightarrow})^m \stackrel{\mu}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^{m'} P'$ and $n = m + m' + 1$. If $s = \ell_1, \ldots, \ell_n$, we call s a *trace*, and we write $P \stackrel{s}{\Longrightarrow}$ if $P \stackrel{\ell_1}{\Longrightarrow} P_1 \stackrel{\ell_2}{\Longrightarrow} P_2 \ldots P_{n-1} \stackrel{\ell_n}{\Longrightarrow} P_n$, for some processes $P_1, \ldots, P_n$. Similarly we write $P \stackrel{s}{\Longrightarrow}_m$ if there are $P_1, \ldots, P_n$ with $P \stackrel{\ell_1}{\Longrightarrow}_{m_1} P_1 \stackrel{\ell_2}{\Longrightarrow}_{m_2} P_2 \ldots P_{n-1} \stackrel{\ell_n}{\Longrightarrow}_{m_n} P_n$, and $m = \Sigma_i m_i$.

Definition 10. *Two processes P, Q of $\mathcal{L}$ are* trace equivalent*, written $P \approx_{\mathrm{tr}} Q$, if for each trace s we have $P \stackrel{s}{\Longrightarrow}$ iff $Q \stackrel{s}{\Longrightarrow}$.*

Two processes P, Q are in the trace equivalence contraction*, written $P \succeq_{\mathrm{tr}} Q$, if, for each trace s:*

1. *if $P \stackrel{s}{\Longrightarrow}_n$ then $Q \stackrel{s}{\Longrightarrow}_m$ for some $m \leq n$;*
2. *if $Q \stackrel{s}{\Longrightarrow}$ then $P \stackrel{s}{\Longrightarrow}$.*

□

Theorem 6. *In CCS, a system of weakly-guarded contractions has a unique solution for $\approx_{\mathrm{tr}}$.* □

In a similar manner, Theorems 4 and 5, concerning divergence, are adapted to trace equivalence. In contrast, both kinds of techniques seem to fail for equivalences allowing infinitary forms of observables. An example is *infinitary trace equivalence*, whereby two processes are equated if they have the same traces, including the infinite ones. For contractions, it is even unclear how the contraction itself for infinitary trace equivalence should be defined. Further, consider the processes $P \stackrel{\mathrm{def}}{=} \Sigma_n a^n$ and $Q \stackrel{\mathrm{def}}{=} P + a.\, K_a.\, \mathbf{0}$, where K_a is the process that can perform infinitely many a actions (i.e., $K_a \stackrel{\triangle}{=} a.\, K_a$): they are not infinitary trace equivalent. However, in an 'infinitary trace' semantics they both are solutions to the (guarded and sequential) contraction

$$X \succeq a + a.\, X$$

The definition of the contraction for infinitary trace equivalence is irrelevant here, because the processes have no τ-transitions. Similarly, we may consider the equation $X = a + a.\, X$, whose syntactic solution has no divergences. The process P above is a solution, yet it is not trace-equivalent to the syntactic solution of the equation, because the syntactic solution has an infinite trace involving a transitions.

5.2 Other languages and results

We refer to [39] and [6] for extensions of the results to languages defined from a generic signature and for conditions based on rule formats for the transition relation of the language. In the same papers, as well as in [8], one may find extensions to calculi with mobility along the lines of the π-calculus. We discuss extensions to higher-order languages in Section 7.2 below.

We refer to the same papers [39, 6, 8] for other results. These include: abstract formulations of the methods; completeness of the methods (the possibility of being

able to use the method to prove all process equalities); the extensions of the results about divergence to behavioural *preorders*; results allowing one to transplant uniqueness of solutions from a system of equations to another one (in the former system divergence may be easier to prove).

6 Formalisation

The paper [44] presents a comprehensive formalisation of Milner's Calculus of Communicating Systems (CCS) in the HOL theorem prover (HOL4), with a focus towards the theory of unique solutions of equations and contractions. Many results in Milner's CCS book [22] are covered, since the unique-solution theorems rely on a large number of fundamental results. Indeed the formalisation encompasses all basic properties of strong, weak bisimilarities (e.g. the fixed-point and substitutivity properties), and their algebraic laws. Further extensions include several versions of "bisimulation up to" techniques, and properties of the expansion and contraction preorders.

The formalisation makes some further refinements to the theory of unique solutions of contractions, in particular concerning the substitutivity problems of weak bisimilarity and other behavioural relations with respect to the sum operator. Thus *rooted bisimilarity* and *rooted contraction*, respectively the coarsest (largest) congruence and precongruence contained in weak bisimilarity and in the contraction preorder when the CCS language includes unguarded sums, are considered. Unique-solution theorems for them are then formulated.

A benefit of the formalisation is that one can take advantage of results about different equivalences and preorders that share similar proof structures. In these cases, when moving between proofs there are only a few places in the proof scripts that have to be modified. The successful termination of a proof gives one the guarantee that the proof is correct, eliminating the risks of overlooking or missing details as in paper-and-pencil proofs.

7 Examples of applications

7.1 Representation of functions as processes

We have used the theory of unique-solutions of equation to study the models produced by representations of functions as processes, precisely encodings of λ-terms under various evaluation strategies, into π-calculus or dialects of it.

In his seminal work [23, 24], Milner shows how the evaluation strategies of *call-by-name λ-calculus* and *call-by-value λ-calculus* [1, 28] can be faithfully mimicked in the π-calculus. More precisely, Milner shows the operational correspondence between reductions in the λ-terms and in the encoding π-terms. He then uses the correspondence to prove that the encodings are *sound*, i.e., if the processes encoding two λ-terms are behaviourally equivalent, then the source λ-terms are also behaviourally equivalent in the λ-calculus. Milner also shows that the converse, *completeness*, fails, intuitively because the encodings allow one to test the λ-terms in all contexts of the π-calculus — more diverse than those of the λ-calculus.

The main problem that Milner's work left open is the characterisation of the equivalence on λ-terms induced by the encoding, whereby two λ-terms are equal if their encodings are behaviourally equivalent π-calculus terms. The question is largely independent of the precise form of behavioural equivalence adopted in the π-calculus because the encodings are deterministic (or at least confluent).

For the call-by-name λ-calculus, the answer was found shortly later [33, 35]: the equality induced is the equality of Levy-Longo Trees (LTs) [19], the lazy variant of Böhm Trees (BTs). It is actually also possible to obtain BTs, by modifying the call-by-name encoding so to allow also reductions underneath a λ-abstraction, and by including divergence among the observables [42].

For call-by-value, in contrast, the problem of identifying the equivalence induced by the encoding has remained open for a long time, for two main reasons. First, tree structures in call-by-value are less studied and less established than in call-by-name. Secondly, existing techniques for reasoning about behaviorual relations in the π-calculus appeared not be powerful enough. For call-by-name, for instance, a central role is played by *bisimulation up-to contexts*. Such a technique, however, cannot be directly applied to Milner's encoding. In [8] the problem was solved by exploiting the theory of unique solution of equations, notably the theory based on divergences. Thus the equivalence induced on λ-terms by their call-by-value encoding into the π-calculus turns out to be *eager normal-form bisimilarity* [16, 17].

In a similar manner, the theory of unique solutions of equations has been used in [32], where the object of study is extensionality in the representation of functions as processes. (In extensional theories two functions are equated if, whenever applied to the same argument, they yield equal results.) Notably, Milner's original encoding of functions as processes is refined into an encoding that is *parametric* on certain abstract components called *wires*. These are, intuitively, processes whose task is to connect two end-point channels. Three main classes of wires are isolated. The first two are dual of each other; the third has a certain parallel behaviour and is the dual of itself. It is shown that the adoption of the parallel wires yields an extensional λ-theory; in fact it yields an equality that coincides with that of BTs with infinite η, shortly $\mathrm{BT}_{\eta\infty}$s. Moreover, the λ-theories produced by the other two classes of wires coincide with the equality of BTs and LTs. All these results rely on the technique of unique-solution of equations (again, in the variant that exploits divergence).

Remark 2. The results above bring up a remarkable agreement between the representation of functions as processes and the classical theory of the λ-calculus. Tree structures play a pivotal role in the the λ-calculus. For instance, trees allow one to unveil the computational content hidden in a λ-term, with respect to some relevant minimal information. In BTs the informations are the head normal forms, whereas in LTs they are the weak head normal forms. Indeed, BTs and LTs produce the same local structures as well-known models of the λ-calculus, such as Plotkin and Scott's P_ω [27, 43], and the *free lazy Plotkin-Scott-Ankle models* [18, 9, 19]. In BTs and LTs the computational content of a λ-term is unveiled using the β-rule alone. In *extensional* structures the β-rule is coupled with the η-rule ($M = \lambda x.\,(Mx)$, for x not free in M). A well-known extensional tree-structure are $\mathrm{BT}_{\eta\infty}$s. The equality

of $BT_{\eta\infty}$s coincides with that of Scott's D_∞ model [43], historically the first model of the untyped λ-calculus. A seminal result by Wadsworth shows that the $BT_{\eta\infty}$s are intimately related to the head normal forms, as the $BT_{\eta\infty}$ equality coincides with contextual equivalence in which the head normal forms are the observables.

7.2 Higher-order languages

As mentioned in the previous section, the unique-solution techniques has been applied and extended to calculi for name mobility in the π-calculus family [39, 6, 8]. More challenging is the transplant of the techniques onto higher-order languages (intuitively, languages where terms of the language are first-order values and may thus instantiate variables), such as the λ-calculi, Higher-order π-calculi, Ambients (e.g., [14, 13, 12, 37, 21])). A major motivation for this application is that unique-solution techniques have the flavour of 'bisimulation up-to context' techniques; that is, techniques in which, during the bisimulation game, the derivatives of two terms can be rewritten so to remove a common context. Up-to context techniques can be particularly effective in higher-order languages. Unfortunately, proving the soundness of the up-to context techniques can be surprisingly hard. Even in pure λ-calculi, and for the most basic form of bisimilarity (e.g., call-by-name or call-by-value λ-calculus and Abramsky's applicative bisimilarity) long-standing open problems remain about soundness. Higher-order process calculi are the class of languages in which up-to context techniques in the literature are most scarce. Recently, techniques of this kind have been derived exploiting fully-abstract translations into first-order calculi (CCS-like or π-calculus-like) [20]. However fully abstract translations are sensitive to the grammar of the language chosen: a modification to the grammar may break or prevent a fully abstract translation to be defined, or, at the very least, will require a careful re-examination of the full abstraction proof. There are however few direct proofs of soundness, that do not rely on translations into first-order languages; e.g., [21, 41] for the Ambient calculus and the Higher-Order π-calculus (HOπ). The Ambient calculus represents a rather special case of higher-order calculus, for processes can move but cannot be communicated. Moving a process is quite different from communicating it as in HOπ: in the former case the process will always be run, immediately and exactly once; in the latter case, in contrast, the process may be copied, and it is the recipient of the process that decides when and where to run each copy. Thus the problems of soundness for up-to context only show up in a limited form in Ambients. The up-to context technique considered in [41] is for environmental bisimilarity. This bisimilarity involves universal quantifications on processes supplied by the environment. Up-to context is essential for limiting the burden due to such quantifications. The contexts used have constraints and are disallowed in certain clauses (which is necessary for the soundness of the technique).

In [7], we have adapted the technique of unique solution of equations (the approach using divergence, as in Section 4) to HOπ using, as a form of bisimilarity, *normal bisimilarity* [34]. Normal bisimilarity has been chosen for two main reasons. First, it is the most effective in proofs, because its clauses do not make use of additional universal quantifications on terms, as other forms of bisimilarity such

as context bisimilarity or environmental bisimilarity. Second, precisely due to such lack of universal quantifications, the 'contextual' properties of normal bisimilarity are quite delicate. Even proving substitutivity with respect to basic operators such as parallel composition is hard (indeed, usually this is proved by relying on mappings onto other forms of bisimilarity or onto first-order calculi). No direct proofs of soundness of forms of up-to contexts exists for normal bisimilarity.

8 Comparison and future work

We briefly compare the method based on contractions and the one based on divergence.

In comparison with the method based on contractions, the main drawback for the method based on divergence is the presence of a semantic condition, involving divergence: the unfoldings of the equations should not produce divergences, or only produce innocuous divergences. Various techniques for checking divergence exist in the literature, including type-based techniques [45, 38, 3]; a syntactic condition is proposed in [4]. 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).

On the other hand, to use contractions for proving an equivalence, one needs also the theory of the associated contraction preorder; moreover there may be processes for which the contraction technique is not applicable simply because the contraction preorder is strictly finer than the equivalence, and therefore one of the processes fails to be a solution.

Formally, the technique based on contraction and the one based on divergence are incomparable. The first can in fact be used also in cases in which the unfolding of the equations produce divergences (plain or innocuous divergences) [6]. 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 they are solutions of the equation $X = a.\, X$; in contrast, only H is a solution of the corresponding contraction.

In practical cases, from what we have seen so far, it seems that the two kinds of technique are equally applicable, and that the amount of work to be carried out is comparable.

More work is needed to properly understand on which behavioural equivalences and which languages the techniques of unique solution of contractions and of equations work. We mentioned that they seem to work if the observables are finitary. More experimentation is needed to formalise appropriate conditions. In particular, we are thinking about higher-order languages, where often up-to-context enhancements of bisimilarity or of other behavioural relations [40] are not directly applicable. Indeed, in these settings often the definition of the behavioural equivalence

or preorder makes use of sophisticated forms of labelled transition systems (e.g., environmental bisimilarity [41]).

Acknowledgments

We are delighted to be able to contribute to the Festschrift in honour of Marjan Sirjani, to whom this paper is dedicated. We would like to take this opportunity for heartily thanking her: both for all her many technical contributions, and for her work in favour of the concurrency of formal methods, including the establishment of the Foundations of Software Engineering (FSEN) conference series, among the most enjoyable conferences that we have had the pleasure of attending.

References

1. Abramsky, S.: The lazy lambda calculus. In: Turner, D. (ed.) Research Topics in Functional Programming, pp. 65–116. Addison-Wesley (1989)
2. Baeten, J., Basten, T., Reniers, M.: Process Algebra: Equational Theories of Communicating Processes. Cambridge University Press (2010)
3. Demangeon, R., Hirschkoff, D., Sangiorgi, D.: Termination in impure concurrent languages. In: Gastin, P., Laroussinie, F. (eds.) Proc. 21th Conf. on Concurrency Theory. Lecture Notes in Computer Science, vol. 6269, pp. 328–342. Springer (2010)
4. Durier, A., Hirschkoff, D., Sangiorgi, D.: Divergence and unique solution of equations. In: Meyer, R., Nestmann, U. (eds.) 28th International Conference on Concurrency Theory, CONCUR 2017. LIPIcs, vol. 85, pp. 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
5. Durier, A., Hirschkoff, D., Sangiorgi, D.: Eager functions as processes. In: 33nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018. IEEE Computer Society (2018)
6. Durier, A., Hirschkoff, D., Sangiorgi, D.: Divergence and unique solution of equations. Logical Methods in Computer Science 15(3) (2019), https://doi.org/10.23638/LMCS-15(3:12)2019
7. Durier, A., Hirschkoff, D., Sangiorgi, D.: Towards 'up to context' reasoning about higher-order processes. Theor. Comput. Sci. 807, 154–168 (2020)
8. Durier, A., Hirschkoff, D., Sangiorgi, D.: Eager functions as processes. Theor. Comput. Sci. 913, 8–42 (2022), https://doi.org/10.1016/j.tcs.2022.01.043
9. Engeler, E.: Algebras and combinators. Algebra Universalis 13, 389–392 (1981)
10. Groote, J.F., Mousavi, M.R.: Modeling and Analysis of Communicating Systems. MIT Press (2014)
11. Hoare, C.: Communicating Sequential Processes. Prentice Hall (1985)
12. Koutavas, V., Wand, M.: Small bisimulations for reasoning about higher-order imperative programs. In: Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 141–152 (2006)
13. Lassen, S.B.: Relational reasoning about contexts. In: Higher-order operational techniques in semantics. pp. 91–135. Cambridge University Press (1998)
14. Lassen, S.B.: Relational Reasoning about Functions and Nondeterminism. Ph.D. thesis, Department of Computer Science, University of Aarhus (1998)
15. Lassen, S.B.: Bisimulation in untyped lambda calculus: Böhm trees and bisimulation up to context. Electr. Notes Theor. Comput. Sci. 20, 346–374 (1999)

16. Lassen, S.B.: Eager normal form bisimulation. In: 20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings. pp. 345–354 (2005)
17. Lassen, S.B., Levy, P.B.: Typed normal form bisimulation. In: Proc. of Computer Science Logic CSL 2007. Lecture Notes in Computer Science, vol. 4646, pp. 283–297. Springer (2007)
18. Lévy, J.: An algebraic interpretation of the $\lambda\beta\kappa$-calculus; and an application of a labelled λ-calculus. Theoretical Computer Science 2(1), 97–114 (1976)
19. Longo, G.: Set theoretical models of lambda calculus: Theory, expansions and isomorphisms. Annales of Pure and Applied Logic 24, 153–188 (1983)
20. Madiot, J., Pous, D., Sangiorgi, D.: Modular coinduction up-to for higher-order languages via first-order transition systems. Log. Methods Comput. Sci. 17(3) (2021)
21. Merro, M., Zappa Nardelli, F.: Behavioral theory for mobile ambients. J. ACM 52(6), 961–1023 (2005)
22. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
23. Milner, R.: Functions as processes. Research Report 1154, INRIA, Sophia Antipolis (1990), Final version in *Journal of Mathem. Structures in Computer Science* 2(2):119–141, 1992
24. Milner, R.: Functions as processes. Journal of Mathematical Structures in Computer Science 2(2), 119–141 (1992)
25. Piérard, A., Sumii, E.: Sound bisimulations for higher-order distributed process calculus. In: Hofmann, M. (ed.) Proc. FOSSACS. Lecture Notes in Computer Science, vol. 6604, pp. 123–137. Springer (2011)
26. Pitts, A.: Howe's method. In: Sangiorgi, D., Rutten, J. (eds.) Advanced Topics in Bisimulation and Coinduction. Cambridge University Press (2012)
27. Plotkin, G.: A set theoretical definition of application. Tech. Rep. Tech. Rep. MIP-R-95, School of A.I., Univ. of Edinburgh (1972)
28. Plotkin, G.: Call by name, call by value and the λ-calculus. Theoretical Computer Science 1, 125–159 (1975)
29. Roscoe, A.W.: An alternative order for the failures model. J. Log. Comput. 2(5), 557–577 (1992)
30. Roscoe, A.W.: The theory and practice of concurrency. Prentice Hall (1998), http://www.cs.ox.ac.uk/people/bill.roscoe/publications/68b.pdf
31. Roscoe, A.W.: Understanding Concurrent Systems. Springer (2010)
32. Sakayori, K., Sangiorgi, D.: Extensional and non-extensional functions as processes. In: 38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023. pp. 1–13. IEEE (2023), https://doi.org/10.1109/LICS56636.2023.10175686
33. Sangiorgi, D.: An investigation into functions as processes. In: Proc. Ninth International Conference on the Mathematical Foundations of Programming Semantics (MFPS'93). Lecture Notes in Computer Science, vol. 802, pp. 143–159. Springer Verlag (1993)
34. Sangiorgi, D.: Bisimulation for Higher-Order Process Calculi. Information and Computation 131(2), 141–178 (1996)
35. Sangiorgi, D.: Lazy functions and mobile processes. In: Plotkin, G., Stirling, C., Tofte, M. (eds.) Proof, Language and Interaction: Essays in Honour of Robin Milner. MIT Press (2000)
36. Sangiorgi, D., Milner, R.: The problem of "Weak Bisimulation up to". In: Cleveland, W. (ed.) Proc. CONCUR '92. Lecture Notes in Computer Science, vol. 630, pp. 32–46. Springer Verlag (1992)

37. Sangiorgi, D., Walker, D.: The π-calculus: a Theory of Mobile Processes. Cambridge University Press (2001)
38. Sangiorgi, D.: Termination of processes. Mathematical Structures in Computer Science 16(1), 1–39 (2006)
39. Sangiorgi, D.: Equations, contractions, and unique solutions. ACM Trans. Comput. Log. 18(1), 4:1–4:30 (2017), http://doi.acm.org/10.1145/2971339
40. Sangiorgi, D.: From enhanced coinduction towards enhanced induction. Proc. ACM Program. Lang. 6(POPL), 1–29 (2022), https://doi.org/10.1145/3498679
41. Sangiorgi, D., Kobayashi, N., Sumii, E.: Environmental bisimulations for higher-order languages. ACM Trans. Program. Lang. Syst. 33(1), 5 (2011)
42. Sangiorgi, D., Xu, X.: Trees from functions as processes. Log. Methods Comput. Sci. 14(3) (2018)
43. Scott, D.S.: Data types as lattices. SIAM J. Comput. 5(3), 522–587 (1976)
44. Tian, C., Sangiorgi, D.: Unique solutions of contractions, ccs, and their HOL formalisation. Inf. Comput. 275, 104606 (2020), https://doi.org/10.1016/j.ic.2020.104606
45. Yoshida, N., Berger, M., Honda, K.: Strong Normalisation in the Pi-Calculus. Information and Computation 191(2), 145–202 (2004)