Equations, contractions, and unique solutions

Davide Sangiorgi

Equations, contractions, and unique solutions

Davide Sangiorgi
Università di Bologna & INRIA
www.cs.unibo.it/~sangio/

Abstract

One of the most studied behavioural equivalences is bisimilarity. Its success is much due to the associated bisimulation proof method, which can be further enhanced by means of 'up-to bisimulation' techniques such as 'up-to context'.

A different proof method is discussed, based on unique solution of special forms of inequations called contractions, and inspired by Milner's theorem on unique solution of equations. The method is as powerful as the bisimulation proof method and its 'up-to context' enhancements. The definition of contraction can be transferred onto other behavioural equivalences, possibly contextual and non-coinductive. This enables a coinductive reasoning style on such equivalences, either by applying the method based on unique solution of contractions, or by injecting appropriate contraction preorders into the bisimulation game.

The techniques are illustrated on CCS-like languages; an example dealing with higher-order languages is also shown.

Categories and Subject Descriptors F.3.1 [*Logics and Meaning of Programs*]: Specifying and Verifying and Reasoning about Programs—logics of programs; F.3.2 [*Logics and Meaning of Programs*]: Semantics of Programming Languages—operational semantics.

General Terms Theory

Keywords Bisimulation; Coinduction; Equations; Unique solution; Contraction

1. Introduction

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

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.

In proofs of bisimilarity results, the bisimulation proof method has become predominant, particularly with the enhancements of the method provided by the so called 'up-to techniques' [26]. Among these, one of the most powerful ones is 'up-to expansion and context', whereby the derivatives of two terms can be rewritten using expansion and bisimilarity and then a common context can be erased. Forms of 'bisimulations up-to context' have been shown to be effective in various fields, including process calculi [24, 26, 34], λ-calculi [14, 16, 17, 35], and automata [7, 30].

The landmark document for bisimilarity is Milner's CCS book [19]. In the book, Milner carefully explains that the bisimulation proof method is not supposed to be the only method for reasoning about bisimilarity. Indeed, various interesting examples in the book are handled using other techniques, notably *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, 28, 29].

Milner's theorem that guarantees unique solutions [19] has however limitations: the equations must be '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, which usually do not include the sum operator).

In this paper we propose a refinement of Milner's technique in which equations are replaced by special inequations called *contractions*. 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 as efficient as P. That is, Q is capable of simulating P by performing less internal work. It is sufficient that Q has one 'efficient' path; Q could also have other paths, that are slower than any path in P. Uniqueness of the solution of a system of contractions is defined as with systems of equations: any two solutions must be equivalent with respect to $\asymp$. The difference with equations is in the meaning of solution: in the case of contractions the solution is evaluated with respect to the preorder $\succeq_{\asymp}$, rather than the equivalence $\asymp$.

If a system of equations has a unique solution, then the corresponding system of contractions, obtained by replacing the equation symbol with the contraction symbol, has a unique solution too. The converse however is false: it may be that only the system of contractions has a unique solution. More important, the condition that guarantees a unique solution in Milner's theorem about equations can be relaxed: 'sequentiality' is not required, and 'guardedness' can be replaced by 'weak guardedness', that is, the variables of the contractions can be underneath *any* prefix, including a prefix representing internal work. (This is the same constraint in Milner's 'unique solution of equations' theorem for *strong* bisimilarity; the constraint is unsound for equations on weak bisimilarity.)

We show that Milner's theorem is not complete for *pure* equations (equations in which recursion is only expressible through the variables of the equations, without using the recursion construct of the process language): there are bisimilar processes that cannot be solutions to the same system of guarded and sequential pure equations. In contrast, completeness holds for weakly-guarded pure contractions. The contraction technique is also *computationally* complete: any bisimulation $\mathcal{R}$ can be transformed into an equivalent system of weakly-guarded contractions that has the same size of $\mathcal{R}$ (where the size of a relation is the number of its pairs, and the size of a system of contractions is the number of its contractions). An analogous result also holds with respect to bisimulation enhancements such as 'bisimulation up-to expansion and context'. The contraction technique is in fact computationally equivalent to the 'bisimulation up-to contraction and context' technique — a refinement of 'bisimulation up-to expansion and context'.

The contraction technique can be generalised to languages whose syntax is the term algebra derived from some signature, and whose semantics is given as an LTS. In this generalisation the weak-guardedness condition for contractions becomes a requirement of *autonomy*, essentially saying that the processes that replace the variables of a contraction do not contribute to the initial action of the resulting expression. The technique can also 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 [9, 10, 22]. For each equivalence, one defines its contraction preorder by controlling the amount of internal work performed.

Finally, we show that a contraction preorder can be injected into the bisimulation game. That is, given an equivalence $\asymp$ and its contraction preorder $\succeq_{\asymp}$, we can define the technique of 'bisimulation up-to $\succeq_{\asymp}$ and context' whereby, in the bisimulation game, the derivatives of the two processes can be manipulated with $\succeq_{\asymp}$ and $\asymp$ (similarly to the manipulations that are possible in the standard 'bisimulation up-to expansion and context' using the expansion relation and bisimilarity) and a common context can then be erased. The resulting 'bisimulation up-to $\succeq_{\asymp}$ and context' is sound for $\asymp$. This technique allows us to derive results for $\asymp$ using the (enhanced) bisimulation proof method.

The contraction technique cannot however be transported onto all (weak) behavioural equivalences. For instance, it does not work in the setting of infinitary trace equivalence (whereby two processes are equal if they have the same finite and infinite traces), and must testing [9]. A discussion on this point is deferred to the concluding section.

We conclude the paper with an example of application of contractions to a higher-order language, which exploits the autonomy condition.

Structure of the paper All background material is reported in Section 2. Contractions and their properties are introduced in Section 3, for bisimilarity and the CCS language. The extension to languages defined from a generic signature is presented in Section 4. The transport of contractions onto other behavioural equivalences is discussed in Sections 5 (barbed congruence), 6 (contextual equivalence), 7 (trace equivalence), and 8 (non-applicability to certain equivalences). The injection of contractions into the bisimulation game is described in Section 9. The example with higher-order languages is reported in Section 10.

2. Background

2.1 CCS

We assume an infinite set of *names* $a, b, \ldots$ and a set of *constant identifiers* (or simply *constants*) for writing recursive 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:

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

where I is a countable indexing set. Sums are guarded so 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 $K \stackrel{\triangle}{=} P$. 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 $K_{\mu.P} \stackrel{\triangle}{=} \mu.\,(P \mid K_{\mu.P})$. The operational semantics is given by means of an LTS, and is reported in Figure 1 (the symmetric version of the two rules for parallel composition has 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, $P \xrightarrow{\widehat{\mu}} P'$ 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 $P = P'$). 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 infix notation for relations, e.g., $P \;\mathcal{R}\; Q$ means that $(P, Q) \in \mathcal{R}$. We use a tilde to denote a tuple, with countably many elements; thus the tuple may also be infinite. All notations are extended to tuples componentwise; e.g., $\widetilde{P} \;\mathcal{R}\; \widetilde{Q}$ means that $P_i \;\mathcal{R}\; Q_i$, for each component i of the tuples $\widetilde{P}$ and $\widetilde{Q}$. And $C[\widetilde{P}]$ is the process obtained by replacing each hole $[\cdot]_i$ of the context C with P_i. We write $\mathcal{R}^{\mathrm{c}}$ for the closure of a relation under contexts. Thus $P \;\mathcal{R}^{\mathrm{c}}\; Q$ means that there are a context C and tuples $\widetilde{P}, \widetilde{Q}$ with $P = C[\widetilde{P}], Q = C[\widetilde{Q}]$ and $\widetilde{P} \;\mathcal{R}\; \widetilde{Q}$. 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, whereas $=$ is used for syntactic equality and for equations). If $\leq$ is a preorder, then $\geq$ is its inverse (and conversely).

2.2 Bisimilarity and expansion

We focus on *weak* behavioural equivalences, which abstract from the number of internal steps performed.

DEFINITION 2.1 (bisimilarity). *A process 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 $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \;\mathcal{R}\; Q'$;*
2. *the converse of (1) on the actions from Q.*

$$\Sigma_{i\in I}\mu_i.\,P_i \xrightarrow{\mu_i} P_i \qquad \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{P \xrightarrow{\mu} P'}{\boldsymbol{\nu} a\, P \xrightarrow{\mu} \boldsymbol{\nu} a\, P'}\ \mu \neq a, \overline{a} \qquad \frac{P \xrightarrow{\mu} P'}{K \xrightarrow{\mu} P'}\ \text{if } K \triangleq P$$

Figure 1. The LTS for CCS

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

We sometimes call bisimilarity *weak* bisimilarity, to distinguish it from *strong bisimilarity*, $\sim$, obtained by replacing in the above definition the weak answer $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ with the strong $Q \xrightarrow{\mu} Q'$.

The bisimulation proof method can be enhanced by means of *up-to techniques*. One of the most useful auxiliary relations in up-to techniques is the *expansion* relation $\succeq_{\rm e}$ [32]. This is an asymmetric version of $\approx$ where $P \succeq_{\rm e} Q$ means that $P \approx Q$, but also that Q achieves the same as P with no more work, i.e. with no more τ actions. Intuitively, if $P \succeq_{\rm e} Q$, we can think of Q as being at least as fast as P or, more generally, we can think that P uses at least as many resources as Q.

DEFINITION 2.2 (expansion). *A process relation $\mathcal{R}$ is an* expansion *if, whenever $P \mathrel{\mathcal{R}} Q$,*

1. *$P \xrightarrow{\mu} P'$ implies that there is Q' with $Q \overset{\widehat{\mu}}{\longrightarrow} Q'$ and $P' \mathrel{\mathcal{R}} Q'$;*
2. *$Q \xrightarrow{\mu} Q'$ implies that there is P' with $P \overset{\mu}{\Longrightarrow} P'$ and $P' \mathrel{\mathcal{R}} Q'$.*

P expands *Q, written $P \succeq_{\rm e} Q$, if $P \mathrel{\mathcal{R}} Q$, for some expansion $\mathcal{R}$.*

Relation $\succeq_{\rm e}$ is studied – using a different terminology – by Arun-Kumar and Hennessy [1]: they show that $\succeq_{\rm e}$ is a mathematically tractable preorder and has a complete proof system for finite terms based on a modification of the standard τ laws for CCS. In CCS, strong and weak bisimilarity are congruence relations, and expansion is a precongruence. It holds that $\sim \subseteq \succeq_{\rm e}$ and $\succeq_{\rm e} \subseteq \approx$; moreover each inclusion is strict. The inclusions are obvious. For the strictness, we have that $P \not\sim \tau.\,P$, $P \preceq_{\rm e} \tau.\,P$, and $\tau.\,P \not\preceq_{\rm e} P$, $\tau.\,P \approx P$.

A powerful up-to technique is 'bisimulation up-to $\succeq_{\rm e}$ and context'. It combines 'up-to expansion' (the possibility of rewriting the derivatives of two related processes using $\succeq_{\rm e}$ and $\approx$), with 'up-to context' (the possibility of removing a common context from the derivatives). We recall that $\mathcal{R}^{\rm c}$ is the context closure of $\mathcal{R}$.

DEFINITION 2.3 (bisimulation up-to $\succeq_{\rm e}$ and context). *A process relation $\mathcal{R}$ is a* bisimulation up-to $\succeq_{\rm e}$ and context *if, whenever $P \mathrel{\mathcal{R}} Q$, we have:*

1. *$P \xrightarrow{\mu} P'$ implies that there is Q' with $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \succeq_{\rm e} \mathcal{R}^{\rm c} \approx Q'$;*
2. *the converse of (1) on the actions from Q.*

The occurrence of $\succeq_{\rm e}$ on the left of $\mathcal{R}^{\rm c}$ cannot be replaced by $\approx$, as this would break the soundness of the technique [26]. The technique is sound [33]:

LEMMA 2.4 (soundness of bisimulation up-to $\succeq_{\rm e}$ and context). *If $\mathcal{R}$ is a bisimulation up-to $\succeq_{\rm e}$ and context, then $R \subseteq \approx$.*

2.3 An example

In examples in the paper, we sometimes use a version of CCS with value passing; this could be translated into pure CCS [19], but having explicit value passing improves readability. 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. The following example illustrates 'bisimulation up-to $\succeq_{\rm e}$ and context', and will then be used for comparison with other techniques.

We wish to implement a server that, when interrogated by clients at a channel c, starts a certain interaction protocol with the client, after consulting an auxiliary server A at a. Here the auxiliary server A is deterministic: at every cycle it outputs an integer value, which changes with the cycle (this change is represented by the successor function, for simplicity).

We consider two implementations of the server. The difference between them is that the first server, L, is lazy, and consults A only *after* a request from a client has been received. In contrast, the other server, E, consults A beforehand so to be then ready in answering a client:

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

Here $R\langle c,x,z\rangle$ 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 actually depend on the values x and z. Process $R\langle c,x,z\rangle$ may also use channel c, and therefore trigger further interactions with the main server; in contrast, $R\langle c,x,z\rangle$ may not use a (i.e., it may not interrogate the auxiliary server).

We use the 'bisimulation up-to expansion and context' technique to prove that the composition of the two servers with A yields bisimilar lazy and eager systems:

$$\begin{array}{rcl} LS\langle n\rangle & \overset{\text{def}}{=} & \boldsymbol{\nu} a\,(A\langle n\rangle \mid L) \\ ES\langle n\rangle & \overset{\text{def}}{=} & \boldsymbol{\nu} a\,(A\langle n\rangle \mid E) \end{array}$$

Relation $\mathcal{R} \overset{\text{def}}{=} \cup_n\{(LS\langle n\rangle, ES\langle n\rangle)\}$ is a bisimulation up-to expansion and context. Consider a pair $(LS\langle n\rangle, ES\langle n\rangle)$. The two processes have one initial transition; the most interesting case is the challenge transition from $ES\langle n\rangle$, and we only consider this one. We have

$$ES\langle n\rangle \xrightarrow{\tau} \boldsymbol{\nu} a\,(A\langle n+1\rangle \mid c(z).\,(E \mid R\langle c,n,z\rangle)) \overset{\text{def}}{=} E'$$

Process $LS\langle n\rangle$ may not produce internal steps, hence its only possible answer is

$$LS\langle n\rangle \Longrightarrow LS\langle n\rangle$$

We can now perform some algebraic manipulations of E': first, we employ CCS expansion law to pull out the prefix at c, then a structural law to resize the scope of the restriction at a in which we exploit the property that $R\langle c,n,z\rangle$ may not use a. (All these laws are valid for strong bisimilarity, hence also for expansion.) We thus obtain:

$$\begin{array}{rcl} E' & \succeq_{\rm e} & c(z).\,(\boldsymbol{\nu} a\,(A\langle n+1\rangle \mid E) \mid R\langle c,n,z\rangle) \\ & = & c(z).\,(ES\langle n+1\rangle \mid R\langle c,n,z\rangle) \overset{\text{def}}{=} E'' \end{array}$$

We can act similarly on $LS\langle n\rangle$, and in addition also employing the law

$$\boldsymbol{\nu} a\,(a(y).\,P \mid \overline{a}\langle v\rangle.\,Q) \approx \boldsymbol{\nu} a\,(P\{^{v}\!/\!_{y}\} \mid Q) \qquad (1)$$

This gives us:

$$\begin{array}{rcl} LS\langle n\rangle & \approx & c(z).\,(\boldsymbol{\nu}a\,(A\langle n+1\rangle \mid L) \mid R\langle c,n,z\rangle) \\ & = & c(z).\,(LS\langle n+1\rangle \mid R\langle c,n,z\rangle) \overset{\text{def}}{=} L' \end{array}$$

We have thus obtained two processes, E'' and L', in the context exclosure of $\mathcal{R}$, and we are done.

In the proof, the 'up-to' techniques allow us to work with a relation that has exactly one pair for each integer. Specifically, 'up-to context' avoids us considering processes in parallel with the lazy and eager systems, whereas 'up-to expansion' allows us to reason only on the 'normal forms' $LS\langle n\rangle$ and $ES\langle n\rangle$ for these systems.

2.4 Systems of equations

Uniqueness of solutions of equations [19] intuitively says that if a context C obeys certain conditions, then all processes P that satisfy the equation $P \approx C[P]$ are bisimilar with each other.

We need variables to write equations. We use capital letters X, Y, Z for these variables and call them *equation variables*. The body of an equation is a CCS expression possibly containing equation variables. Thus such expressions, ranged over by E, live in a CCS grammar extended with equation variables.

DEFINITION 2.5. *Assume that, for each i of a countable indexing set I, we have variables X_i, and expressions E_i possibly containing such variables. Then*

$$\{X_i = E_i\}_{i\in I}$$

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, assuming $\widetilde{P}$ and $\widetilde{X}$ have the same length. (This is syntactic replacement, akin to the substitution of the holes of a context with processes.) The components of $\widetilde{P}$ need not be different from each other, as it must be for the variables $\widetilde{X}$. If the system has infinitely many equations, the tuples $\widetilde{P}$ and $\widetilde{X}$ are infinite too.

DEFINITION 2.6. *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 $\widetilde{P} \approx \widetilde{Q}$.*

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 $K \overset{\triangle}{=} a.\,K$: for any other solution P we have $P \approx K$. The unique solution of (2), modulo $\approx$, are the constants K_1, K_2 with $K_1 \overset{\triangle}{=} a.\,K_2$ and $K_2 \overset{\triangle}{=} b.\,K_1$; 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 $K \overset{\triangle}{=} a.\,K$.

DEFINITION 2.7. *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;*
- 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 subexpression 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, 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.

THEOREM 2.8 (unique solution of equations, [19]). *A system of guarded and sequential equations has a unique solution for $\approx$.*

The proof exploits an invariance property on immediate transitions for 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 [19])

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

where X is guarded but not sequential. Any processes that does not use a is a solution.

3. Contractions

In Theorem 2.8 the constraints on guardedness and, especially, on sequentiality limit its applicability. Further, the same definitions and examples discussed for bisimilarity (and hence also the same limitations) apply to other behavioural equivalences; e.g., contextual equivalence and trace-based equivalences.

One may wonder if the conditions of Theorem 2.8 can be relaxed by simply requiring that each equation be *sequentially guarded*, that is, of the form $X = \Sigma_j \ell_j.\,E_j$ (where ℓ_j is a visible action). Unfortunately, uniqueness still fails; a counterexample is

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

Any process P with $P \approx a.\,P'$, and P' unable to use a, is a solution. Examples are $a.\,\mathbf{0}$ and $a.\,b.\,\mathbf{0}$.

An equation $X = a.\,E$ need not have a unique solution even if we confine ourselves to processes that may only perform a transitions. An example is the equation

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

Here the body of the equation produces an a, cancels the first a from X and then reproduces all other a's. Any process P with $P \approx a.\,P'$ for some P', is a solution; for instance, $a.\,\mathbf{0}$ or $a.\,a.\,\mathbf{0}$ or even $!a.\,\mathbf{0}$.

3.1 Contraction preorders

The constraints on the unique-solution Theorem 2.8 can be weakened if we move from equations to certain inequations that we call *contractions*.

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 then 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 first explain the idea on the concrete case of bisimilarity.

DEFINITION 3.1 (bisimulation contraction). *A process relation $\mathcal{R}$ is a* bisimulation contraction *if, whenever $P \mathrel{\mathcal{R}} Q$,*

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

Bisimilarity contraction*, written $\succeq_{\rm bis}$, 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 3.1, on the challenges from Q, entirely ignores efficiency: it is the same clause of weak bisimulation — the final derivatives are even required to be related by $\approx$, rather than by $\mathcal{R}$.

Bisimilarity contraction is coarser than the expansion relation $\succeq_{\rm e}$ of Definition 2.2. Clause (1) is the same in the two definitions. But in clause (2) expansion uses $P \overset{\mu}{\Longrightarrow} P'$, rather than $P \overset{\widehat{\mu}}{\Longrightarrow} P'$; moreover with contraction the final derivatives are simply required to be bisimilar. An expansion $P \succeq_{\rm e} Q$ tells us that Q is always more efficient than P, whereas the contraction $P \succeq_{\rm bis} Q$ just says that Q has the possibility of being more efficient than P.

EXAMPLE 3.2. We have $a \not\succeq_{\rm bis} \tau.\, a$. However, $a + \tau.\, a \succeq_{\rm bis} a$, as well as its converse, $a \succeq_{\rm bis} a + \tau.\, a$. Indeed, if $P \approx Q$ then $P \succeq_{\rm bis} P + Q$. The last two relations do not hold with $\succeq_{\rm e}$, which explains the strictness of the inclusion $\succeq_{\rm e} \subseteq \succeq_{\rm bis}$. $\square$

As bisimilarity contraction follows expansion in one direction and bisimilarity in the other, clearly separating the two, the precongruence and congruence for such relations can be combined into a precongruence proof for the contraction.

THEOREM 3.3. *$\succeq_{\rm bis}$ is a precongruence in CCS.*

3.2 Systems of contractions

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 3.4. *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 $\succeq_{\rm bis}$, and the equivalence $\asymp$ as the bisimilarity $\approx$. Thus $\widetilde{P}$ solution for $\succeq_{\rm bis}$ of the system of contractions $\{X_i \succeq E_i\}_{i\in I}$ means that $\widetilde{P} \succeq_{\rm bis} \widetilde{E}[\widetilde{P}]$; and the system having a unique solution for $\approx$ means that whenever $\widetilde{P}$ and $\widetilde{Q}$ are both solutions for $\succeq_{\rm bis}$ then $\widetilde{P} \approx \widetilde{Q}$.

LEMMA 3.5. *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 lemma holds because any system of equations has at least one solution for strong bisimilarity, obtained by interpreting the equations as recursive process definitions. A solution for strong bisimilarity is also a solution in the system of contractions. Moreover, any solution of the system of contractions is a solution of the system of equations. (The converse, in contrast, is false: as we shall see, systems of contractions more easily have a unique solution.)

3.3 Conditions for unique solution of contractions

DEFINITION 3.6. *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.*

In proofs about weakly-guarded contractions we will often unfold the contractions, exploiting the substitutivity of the contraction preorder, with the objective of placing processes that are solutions of the contractions underneath a certain number of prefixes. Suppose $\widetilde{P}$ are solutions of a system of contractions $\{X_i \succeq E_i\}_{i\in I}$, and consider a context $C[\widetilde{P}]$. Then the process obtained from $C[\widetilde{P}]$ by unfolding the contractions once is $C[\,\widetilde{E}[\widetilde{P}]\,]$; the process obtained by unfolding the contractions twice is $C[\,\widetilde{E}[\widetilde{E}[\widetilde{P}]]\,]$; and similarly for the n-unfolding.

LEMMA 3.7. *Suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\approx$ of a system of weakly-guarded contractions. For any context C, if $C[\widetilde{P}] \overset{\mu}{\Longrightarrow} R$, then there is a context C' such that $R \succeq_{\rm bis} C'[\widetilde{P}]$ and $C[Q] \overset{\widehat{\mu}}{\Longrightarrow}\approx C'[\widetilde{Q}]$.*

Proof [Sketch] Let n be the length of the transition $C[\widetilde{P}] \overset{\mu}{\Longrightarrow} R$ (the number of 'strong steps' of which it is composed), and let $C''[\widetilde{P}]$ and $C''[\widetilde{Q}]$ be the processes obtained from $C[\widetilde{P}]$ and $C[\widetilde{Q}]$ by unfolding the definitions of the contractions n times. Thus in C'' each hole is underneath at least n prefixes, and cannot contribute to an action in the first n transitions.

We have $C[\widetilde{P}] \succeq_{\rm bis} C''[\widetilde{P}]$, and $C[\widetilde{Q}] \succeq_{\rm bis} C''[\widetilde{Q}]$, by the precongruence properties of $\succeq_{\rm bis}$. Moreover, since each hole of the context C'' is underneath at least n prefixes, applying the definition of $\succeq_{\rm bis}$ on the transition $C[\widetilde{P}] \overset{\mu}{\Longrightarrow} R$, we infer the existence of C' such that $C''[\widetilde{P}] \overset{\widehat{\mu}}{\Longrightarrow} C'[\widetilde{P}] \preceq_{\rm bis} R$ and $C''[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow} C'[\widetilde{Q}]$. Finally, again applying the definition of $\succeq_{\rm bis}$ on $C[\widetilde{Q}] \succeq_{\rm bis} C''[\widetilde{Q}]$, we derive $C[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow}\approx C'[\widetilde{Q}]$. $\square$

THEOREM 3.8 (unique solution of contractions for $\approx$). *A system of weakly-guarded contractions has a unique solution for $\approx$.*

Proof [Sketch] One shows that if $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\approx$ of a system of weakly-guarded contractions, then the relation

$$\mathcal{R} \overset{\text{def}}{=} \{(R,S) \mid R \approx C[\widetilde{P}], S \approx C[\widetilde{Q}] \text{ for some context } C\}\,.$$

is a bisimulation, exploiting Lemma 3.7. $\square$

In comparison to Theorem 2.8 for equations, in Theorem 3.8 for contractions the 'guardedness' condition is weakened, allowing variables that are underneath τ prefixes; most important, the sequentiality condition is removed, allowing variables underneath any process constructs.

EXAMPLE 3.9. The following contractions have a unique solution for $\approx$:

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

We have seen in Section 2.4 and at the beginning of Section 3 that 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. It is easy to see that an inactive process is solution. Conversely, suppose P is not inactive, and let n be the least $n \geq 0$ such that $P(\xrightarrow{\tau})^n \xrightarrow{\ell}$ for some ℓ; then P is not a solution of $P \succeq \tau.\, P$ because $\tau.\, P$ needs at least $n+1$ τ-steps before exhibiting any visible action, and therefore can never be more efficient than P. Example of solutions for (2) and (3) are $a.\, P$ and $\tau.\, a.\, P$, where P is inactive. Any solution of (2) and (3) is bisimilar with $a.\, \mathbf{0}$. □

REMARK 3.10. Results such as Lemma 3.7 and Theorem 3.8 also hold if the game played in clause (1) of Definition 3.1 of bisimulation contraction is that of strong simulation (i.e., "$P \xrightarrow{\mu} P'$ implies there is Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \mathcal{R} Q'$"). However, the resulting relation would not be coarse enough to capture expansion – a major goal for this paper is understanding existing 'bisimilarity up-to' techniques, where expansion is important. □

3.4 Completeness

An interesting class of contractions are those in which the body E of each contraction $X \succeq E$ does not contain constants. In these systems, all forms of infinity in the behaviour of processes are captured by recursive calls through the contraction variables. We call such systems *pure*. Similarly we call pure a system of equations without constants. In this section we discuss the expressiveness of pure systems of contractions and equations. (With constants the question is vacuous, as the behaviour of any process P is captured by the guarded and sequential equation $X = P$.) We show that the technique of weakly-guarded contractions given by Theorem 3.8 is complete, whereas that of guarded and sequential equations given by Theorem 2.8 is not.

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 ($i = 1, 2$).

THEOREM 3.11 (completeness). *Suppose $\mathcal{R}$ is a bisimulation. Then there is a system of weakly-guarded pure contractions of which $\mathcal{R}_1$ and $\mathcal{R}_2$ are solutions for $\succeq_{\mathrm{bis}}$.*

Proof [Sketch] Suppose $\mathcal{R}$ is a bisimulation. We define a system of contractions of which $\mathcal{R}_1$ and $\mathcal{R}_2$ are solutions. The variables of the contractions are of the form $X_{P,Q}$ for $P \mathcal{R} Q$, and there is one contraction for each pair in $\mathcal{R}$.

We show how the contraction for a pair $P \mathcal{R} Q$ is built. Consider an enumeration of all the transitions from P:

$$P \xrightarrow{\mu_r} P_r$$

where r ranges over some countable set I_P. Following the bisimulation game, for each r there is Q_r s.t. $Q \overset{\widehat{\mu_r}}{\Longrightarrow} Q_r$ and $P_r \mathcal{R} Q_r$. Proceeding similarly on the challenge transitions from Q, i.e. $Q \xrightarrow{\mu_s} Q_s$ for $s \in I_Q$, we find processes P_s with $P \overset{\widehat{\mu_s}}{\Longrightarrow} P_s$ and $P_s \mathcal{R} Q_s$. Then the contraction for the pair P, Q is:

$$X_{P,Q} \succeq \Sigma_r \mu_r.\, X_{P_r,Q_r} + \Sigma_s \mu_s.\, X_{P_s,Q_s}$$

□

The contractions in the proof of the theorem are sequential and weakly guarded, but not necessarily guarded. The assertion of Theorem 3.11 can actually be refined: the technique based on weakly-guarded contractions is also *computationally complete* with respect to the bisimulation proof method, in the sense that the size of the structures needed and the subsequent amount of checks are comparable. The *size* of a relation is the number of its pairs. The size of a system of contractions is the number of contractions. The proof of Theorem 3.11 shows that the system of contractions derived from a bisimulation $\mathcal{R}$ has the same size as $\mathcal{R}$; moreover, the work needed to prove that $\mathcal{R}_1$ and $\mathcal{R}_2$ are solutions of the system of contractions is precisely the work needed to check the challenge/response diagrams of the bisimulation game for $\mathcal{R}$.

In contrast, the method for equations resulting from Theorem 2.8 is not complete. For instance, there is no system of guarded and sequential pure equations in which one of the solutions is the process K so defined:

$$K \overset{\triangle}{=} \tau.\, (a \mid K) + \tau.\, \mathbf{0}\ .$$

To see this, it is useful to express the behaviour of K via the following constants:

$$\begin{array}{rcl} H_0 & \overset{\triangle}{=} & \tau.\, H_1 + \tau \\ H_i & \overset{\triangle}{=} & \tau.\, H_{i+1} + a.\, H_{i-1} + \tau.\, a^i \quad (i > 0) \end{array}$$

We have $a^i \mid K \approx H_i$, for each i, as witnessed by the relation

$$\mathcal{R} \overset{\text{def}}{=} \cup_{i \geq 0} \{(a^i \mid K, H_i)\}\ ,$$

which is a bisimulation up-to strong bisimilarity. Now, for each $n \neq m$, we have $H_n \not\approx H_m$ (because, assuming $n < m$, H_m cannot match the transition $H_n \xrightarrow{\tau} a^n$); moreover, for each n there is a transition $H_n \xrightarrow{\tau} H_{n+1}$. As a consequence, the infinite sequence of transitions

$$H_0 \xrightarrow{\tau} H_1 \xrightarrow{\tau} \cdots H_n \xrightarrow{\tau} H_{n+1} \xrightarrow{\tau} \cdots \tag{2}$$

go through states that are pairwise non-bisimilar. An equation of which K is solution should be able to express the same behaviour. This is impossible, however, if the equation is sequential and guarded, because the equation variables must be underneath a visible prefix, and can only be reached by performing a visible action. Hence an infinite nesting of internal transitions as in (2) cannot be derived.

3.5 Relationship with up-to context

The completeness of the contraction technique given by Theorem 3.11, including the computational completeness discussed after the theorem, remains also with respect to powerful enhancements of the bisimulation proof method such as 'up-to context' techniques.

We show that the contraction technique is in fact computationally equivalent to the 'up-to $\succeq_{\mathrm{bis}}$ and context' technique, a refinement of the 'up-to expansion and context' of Definition 2.3 (the former captures a larger set of relations because bisimilarity contraction is coarser than expansion).

DEFINITION 3.12 (bisimulation up-to $\succeq_{\mathrm{bis}}$ and context). *A process relation $\mathcal{R}$ is a* bisimulation up-to $\succeq_{\mathrm{bis}}$ and context *if, whenever $P \mathcal{R} Q$, we have:*

1. *$P \xrightarrow{\mu} P'$ implies there is Q' such that $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \succeq_{\mathrm{bis}} \mathcal{R}^{\mathrm{c}} \approx Q'$*
2. *the converse of (1) on the actions from Q.*

THEOREM 3.13. *Suppose $\mathcal{R}$ is a bisimulation up-to $\succeq_{\mathrm{bis}}$ and context. Then there is a system of weakly-guarded contractions, of the same size as $\mathcal{R}$, of which $\mathcal{R}_1$ and $\mathcal{R}_2$ are solutions for $\succeq_{\mathrm{bis}}$.*

Conversely, suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\succeq_{\rm bis}$ to the same system of weakly-guarded contractions. Then the relation $\{(P_i, Q_i)\}_i$ is a bisimulation up-to $\succeq_{\rm bis}$ and context.

The theorem is proved along the lines of Theorem 3.11.

REMARK 3.14. The definition of contraction was derived by attempts at obtaining theorems such as 3.11 and 3.13. The constructions in the theorems do not work with expansion in place of bisimilarity contraction. □

From Theorems 3.13 and 3.8 we derive:

COROLLARY 3.15. (soundness of 'bisimulation up-to $\succeq_{\rm bis}$ and context'). *If $\mathcal{R}$ is a bisimulation up-to $\succeq_{\rm bis}$ and context, then $R \subseteq \approx$.*

Having shown that the techniques of weakly-guarded contractions and 'bisimulation up-to $\succeq_{\rm bis}$ and context' are equivalent, we can derive the soundness of one from the soundness of the other (in Corollary 3.15 we took the contraction technique as primitive). The complexity of the soundness proofs of the two techniques is similar. The main difference is that the expressions in the body of the contractions are weakly guarded, whereas the contexts of the 'up-to context' bisimulation techniques are not. As a consequence, in the proofs for the 'up-to context' techniques one has to reason about all possible interactions between a context and the processes plugged into it, proceeding by transition induction and a case analysis on the last rule used to derive a transition. With contractions this is avoided, exploiting the weak-guardedness condition and the unfolding of the contractions.

3.6 Example: the lazy and eager servers

We show a proof of the bisimilarity between the lazy and the eager systems of Section 2.3 using the technique of unique solution of contractions. This serves both as an an illustration of the application of the technique, and as a comparison with the technique based on the bisimulation proof method employed in the proof of Section 2.3.

The proof consists in showing that $\{LS\langle n\rangle\}_n$ and $\{ES\langle n\rangle\}_n$ are solutions of the following system of contractions:

$$\{X_n \succeq c(z).\,(X_{n+1} \mid R\langle c,n,z\rangle)\}_n \qquad (3)$$

We establish that $\{ES\langle n\rangle\}_n$ is a solution using the algebraic laws in the proof in Section 2.3 (noticing that in (1) $\approx$ can be replaced by $\succeq_{\rm bis}$):

$$\begin{array}{rcl} ES\langle n\rangle & \succeq_{\rm bis} & \boldsymbol{\nu}a\,(A\langle n+1\rangle \mid c(z).\,(E \mid R\langle c,n,z\rangle)) \\ & \sim & c(z).\,(\boldsymbol{\nu}a\,(A\langle n+1\rangle \mid E) \mid R\langle c,n,z\rangle) \\ & = & c(z).\,(ES\langle n+1\rangle \mid R\langle c,n,z\rangle)\ . \end{array}$$

We proceed similarly for $LS\langle n\rangle$.

The contraction (3) is not sequential, hence contractions and Theorem 3.8 cannot be replaced by equations and Theorem 2.8.

4. Language generalisation

We have shown the property of unique solution of weakly-guarded contractions in CCS. We generalise here the theorem to an arbitrary process language, using a more abstract condition. The generalisation serves both to better understand the validity of the theorem, and for applicability to languages that, unlike CCS, do not have an explicit prefixing construct.

For this generalisation we consider the case — standard in process algebra — in which the syntax of the processes is the term algebra generated by some signature, and the semantics is given as an LTS. We call *process language* any such language. We use $\mathcal{L}$ to denote a generic process language, and $\mathcal{L}(\mathcal{X})$ for its extension with the contraction variables in $\mathcal{X}$. We say that $\mathcal{L}$ is *$\approx$-safe* if, in $\mathcal{L}$, $\approx$ is a congruence relation, and its corresponding contraction $\succeq_{\rm bis}$ is a precongruence.

In Theorem 3.8, the 'weakly guarded' hypothesis makes sure that the body of a contraction alone determines the first interaction. The body is thus autonomous: the interaction occurs without contributions from the terms that replace the contraction variables. Whenever the bodies of the contractions are autonomous in this sense, the unique-solution property holds.

DEFINITION 4.1 (autonomous contractions). *An expression E of $\mathcal{L}(\mathcal{X})$ is* autonomous *if for all processes $\widetilde{P}$ of $\mathcal{L}$ we have:*

- *if $E[\widetilde{P}] \xrightarrow{\mu} R$, then there is a context C such that $R = C[\widetilde{P}]$, and for all $\widetilde{Q}$, also $E[\widetilde{Q}] \xrightarrow{\mu} C[\widetilde{Q}]$.*

A system of contractions $\{X_i \succeq E_i\}_{i\in I}$ is autonomous *if each expression E_i is autonomous.*

THEOREM 4.2. *In any $\approx$-safe process language $\mathcal{L}$, a system of autonomous contractions has a unique solution for $\approx$.*

The proof of the theorem is similar to that of Theorem 3.8. Checking the autonomy property is often straightforward. For instance, in the case of the *GSOS* format [5], autonomy holds if, in the body E of a contraction, all variables are underneath an axiom operator, that is, an operator that, as CCS prefix, is defined by means of SOS rules in which the set of hypothesis is empty.

In some cases, autonomy may be better than the weakly-guarded hypothesis even if the calculus has a prefix operator: we shall see an example in Section 10, with the Higher-Order π-calculus, where autonomy allows us to capture occurrences of the contraction variables *within* output prefixes.

5. Barbed congruence

We (briefly) consider the application of the idea of contraction to barbed congruence [20], for various reasons. First, barbed congruence is a contextually-defined form of behavioural equivalence, and it paves the way to the treatment of other forms of contextual equivalence. Second, we want to show that – sometimes – the contraction techniques make it possible to work directly with barbed congruence, even though it is contextually defined (e.g., the example with higher-order processes in Section 10). Third, the definition of barbed congruence applies to any language with a reduction semantics (i.e., a *reduction* relation and a *barb*, or *observation*, predicate), as opposed to the LTS semantics of the languages in earlier sections.

Thus the definitions and results in this section hold for any algebraic calculus (the term algebra over a signature) equipped with a reduction semantics, that is, a reduction relation $\longrightarrow$ and a barb predicate $\downarrow$. We use $\mathcal{RL}$ for referring to a generic such language, and $\mathcal{RL}(\mathcal{X})$ for its extension with the contraction variables in $\mathcal{X}$. (For CCS, $\longrightarrow$ is $\xrightarrow{\tau}$ and $P\downarrow$ holds if $P \xrightarrow{\ell}$, for some visible action ℓ.) As usual, $\Longrightarrow$ is the reflexive and transitive closure of $\longrightarrow$; and $P \Downarrow$ holds if there is P' with $P \Longrightarrow P'$ and $P'\downarrow$.

DEFINITION 5.1 (barbed bisimulation and congruence). *A relation $\mathcal{R}$ on the processes of $\mathcal{RL}$ is a* barbed bisimulation *if whenever $P\;\mathcal{R}\;Q$:*

1. *$P \longrightarrow P'$ implies there is Q' such that $Q \Longrightarrow Q'$ and $P'\;\mathcal{R}\;Q'$;*
2. *$P\downarrow$ implies $Q\Downarrow$;*
3. *the converse of (1) and (2), on challenges from Q.*

Barbed bisimilarity, written $\approx_{\rm bar}$, is the union of all barbed bisimulations. Two processes P and Q are barbed congruent*, written*

$P \approx^{\mathrm{c}}_{\mathrm{bar}} Q$*, if for each context* C*, it holds that* $C[P] \approx_{\mathrm{bar}} C[Q]$. □

REMARK 5.2. The definitions of barbed congruence in the literature often make use of a *set* of barb predicates; we use only *one* barb here for mere simplicity of presentation. A variant of barbed congruence is *reduction-closed barbed congruence* [11], in which the closure under contexts is placed within the definition of bisimilarity. The difference between the two variants has no consequences on the results in the paper. □

In the 'contraction version' of barbed bisimilarity we write $Q \xrightarrow{\wedge} Q'$ if $Q \longrightarrow Q'$ or $Q = Q'$.

DEFINITION 5.3. (barbed contraction, barbed congruence contraction). *A relation* $\mathcal{R}$ *on the processes of* $\mathcal{RL}$ *is a* barbed contraction *if, whenever* $P \mathrel{\mathcal{R}} Q$*:*

1. $P \longrightarrow P'$ *implies there is* Q' *such that* $Q \xrightarrow{\wedge} Q'$ *and* $P' \mathrel{\mathcal{R}} Q'$*;*
2. $Q \longrightarrow Q'$ *implies there is* P' *such that* $P \Longrightarrow P'$ *and* $P' \approx_{\mathrm{bar}} Q'$*;*
3. $P \downarrow$ *implies* $Q \downarrow$*;*
4. $Q \downarrow$ *implies* $P \Downarrow$*.*

Barbed contraction*, written* $\succeq_{\mathrm{bar}}$*, is the union of all barbed contractions.* Barbed congruence contraction*, written* $\succeq^{\mathrm{c}}_{\mathrm{bar}}$*, relates two processes* P *and* Q *if, for each context* C*, it holds that* $C[P] \succeq_{\mathrm{bar}} C[Q]$.

We transport the concept of autonomy to reduction-based semantics.

DEFINITION 5.4 (reduction-autonomous contractions). *An expression* E *of* $\mathcal{RL}(\mathcal{X})$ *is* reduction-autonomous *if for all processes* $\widetilde{P}$ *and context* C *of* $\mathcal{RL}$*:*

- *if* $C[E[\widetilde{P}]] \longrightarrow R$*, then there is a context* C' *such that* $R = C'[\widetilde{P}]$ *and, for all* $\widetilde{Q}$*, also* $C[E[\widetilde{Q}]] \longrightarrow C'[\widetilde{Q}]$*;*
- *if* $C[E[\widetilde{P}]] \downarrow$ *then, for all* $\widetilde{Q}$*, also* $C[E[\widetilde{Q}]] \downarrow$*.*

A system of contractions $\{X_i \succeq E_i\}_{i \in I}$ *is* reduction-autonomous *is each expression* E_i *is reduction-autonomous.*

Barbed congruence and its contraction are, by definition, fully substitutive. Hence the safety requirement of Theorem 4.2 is not needed. In the property of unique solution for barbed congruence, the symbols $\asymp$ and $\succeq_{\asymp}$ of Definition 3.4 become $\approx^{\mathrm{c}}_{\mathrm{bar}}$ and $\succeq^{\mathrm{c}}_{\mathrm{bar}}$, respectively.

THEOREM 5.5. *In* $\mathcal{RL}$*, any system of reduction-autonomous contractions has a unique solution for* $\approx^{\mathrm{c}}_{\mathrm{bar}}$*.*

6. Uniqueness of solution of contractions for non-coinductive equivalences

We consider now non-coinductive equivalences. We focus on contextual equivalence [22] (i.e., *may testing* [9]), because it is widely studied. As the barbed congruence of Section 5, so contextual equivalence is contextually defined. Thus the setting considered is the same: an algebraic process language equipped with a reduction semantics. We reuse notations and terminologies from Section 5. Intuitively, two terms are contextually equivalent when they are equally observable, in any context.

DEFINITION 6.1 (contextual equivalence). $P \approx_{\mathrm{ctx}} Q$ *holds when* $C[P] \Downarrow$ *iff* $C[Q] \Downarrow$*, for all* C*.*

The definition of the contextual equivalence contraction uses the predicate $P \Downarrow^n$, indicating that a barb is reached in at most n steps, i.e., $P(\xrightarrow{\tau})^m P' \downarrow$, for some P', with $0 \le m \le n$.

DEFINITION 6.2 (contextual equivalence contraction). $P \succeq_{\mathrm{ctx}} Q$ *if for all* C*:*

1. $C[P] \Downarrow^n$ *implies* $C[Q] \Downarrow^n$*, for any* n*;*
2. $C[Q] \Downarrow$ *implies* $C[P] \Downarrow$*.*

Thus, referring to contextual equivalence, the symbols $\asymp$ and $\succeq_{\asymp}$ of Definition 3.4 become $\approx_{\mathrm{ctx}}$ and $\succeq_{\mathrm{ctx}}$.

THEOREM 6.3. *A system of reduction-autonomous contractions has a unique solution for* $\approx_{\mathrm{ctx}}$*.*

Proof [Sketch] Suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\succeq_{\mathrm{ctx}}$, and consider a context C. One shows that $C[\widetilde{P}] \Downarrow$ implies $C[\widetilde{Q}] \Downarrow$. Suppose $C[\widetilde{P}] \Downarrow^n$. One proceeds by induction on n, exploiting the properties of the unfolding of contractions. □

COROLLARY 6.4. *In CCS, a system of weakly-guarded contractions has a unique solution for* $\approx_{\mathrm{ctx}}$*.*

6.1 Example: the lazy and eager servers, revisited

Contextual equivalence does not have the congruence problems of bisimilarity for summation that motivated, in the presentation of CCS in Section 2, the use of guarded sums. We can therefore admit the full summation construct $\Sigma_i P_i$ in the grammar for the CCS processes. Such a flexibility will be useful in this example.

We revisit the lazy and eager servers in the example of Section 2.3. We modify the auxiliary server A, which was consulted by the main server before starting an interaction protocol with a client. In Section 2.3, the server was deterministic; now it is nondeterministic. Thus all definitions remain the same except that A always returns an arbitrary integer:

$$A \overset{\triangle}{=} \Sigma_{n \in N} \ \overline{a}\langle n \rangle .\, A$$

The system with the lazy main server is now $LS \overset{\mathrm{def}}{=} \boldsymbol{\nu} a\,(A \mid L)$, and the system with the eager main server is $ES \overset{\mathrm{def}}{=} \boldsymbol{\nu} a\,(A \mid E)$, where L and E are as in Section 2.3. The timing difference between LS and ES in consulting A is observable under bisimilarity. The reason is that an interaction

$$ES \longrightarrow \boldsymbol{\nu} a\,(A \mid c(z).\,(E \mid R\langle c, n, z\rangle))\,,$$

in which n is received from A, is a commitment to using n in the interaction with the next client. In contrast, LS is unable to make such a commitment — its only initial transition is a visible one.

The difference is however not observable under contextual equivalence. We prove $LS \approx_{\mathrm{ctx}} ES$ using the technique of unique solution of weakly-guarded contractions. The proof is similar to that with the deterministic auxiliary server and the bisimilarity contraction in Section 3.6, with a further simplification: a single contraction is sufficient, namely

$$X \succeq c(z).\, \Sigma_n (X \mid R\langle c, n, z\rangle) \tag{4}$$

To show that both LS and ES are solutions for $\succeq_{\mathrm{ctx}}$ of this contraction, we employ the same laws and algebraic reasoning of Section 3.6 (which are sound because bisimilarity implies contextual equivalence). An additional laws is required in the proof of ES:

$$\Sigma_i \alpha.\, R_i \succeq_{\mathrm{ctx}} \alpha.\, \Sigma_i R_i \quad (\alpha \text{ is any prefix}) \tag{5}$$

(the law is actually valid for *strong* contextual equivalence, where two equal processes are required to reach an observable in the same number of steps). This is one of the most distinguishing laws of contextual equivalence. Using the laws we have:

$$\begin{array}{rcl} ES = \boldsymbol{\nu} a\, (A \mid E) & \succeq_{\mathrm{ctx}} & \Sigma_n c(z).\, \boldsymbol{\nu} a\, (A \mid E \mid R\langle c, n, z\rangle) \\ & \succeq_{\mathrm{ctx}} & c(z).\, \Sigma_n (\boldsymbol{\nu} a\, (A \mid E) \mid R\langle c, n, z\rangle) \\ & = & c(z).\, \Sigma_n (ES \mid R\langle c, n, z\rangle) \end{array}$$

which shows that ES is a solution of the contraction (4). The proof that also LS is a solution is simpler.

The above proof is similar to the proofs with the servers in Sections 2.3 and 3.6. All these proofs, explicitly or implicitly, employ 'up-to context' reasoning; above the common context is $c(z).\, \Sigma_n([\cdot] \mid R\langle c, n, z\rangle)$.

A proof that follows the definition of contextual equivalence would be hard due to the quantification on all contexts. In CCS, contextual equivalence coincides with trace equivalence. The equality in the example cannot be proved purely algebraically, using standard axiom systems for trace equivalence, because the systems compared are not finite or finite state (axiomatisations are complete only on these systems). One could show that ES and LS have the same traces proceeding by induction on the length of the traces. The proof is tedious, for instance because R could be any process.

REMARK 6.5. The proof of the equality between ES and LS reveals the essence of the technique based on unique solution of contractions. One employs some simple algebraic laws to prove that two tuples of terms are solutions of a certain system of contractions, from which the equality between the two tuples is derived from the unique-solution theorem. The algebraic laws may have been obtained in various ways (e.g., an axiomatisation of the equality for the finite terms). In our example we have used laws for bisimilarity, because it implies contextual equivalence, plus law (5). Law (5) is a well-known law in axiomatisations of trace equivalence; the law can also be easily proved directly in terms of contextual equivalence, reasoning by induction on the length on the number of τ-steps needed to reach an observable. ☐

7. Trace equivalence

In this section we briefly consider trace equivalence. In CCS and most process algebras, trace equivalence is a direct characterisation of contextual equivalence, in the same way as bisimilarity is for barbed congruence. Aside from this, we look at trace equivalence because it is a behavioural equivalence non-contextual and inductive. We use s to range over *traces*, i.e., non-empty sequences of visible actions.

We assume to be in a generic process language $\mathcal{L}$ with an LTS semantics, as in Section 4. We write $P \overset{\mu}{\Longrightarrow}_n P'$ if $P \overset{\mu}{\Longrightarrow} P'$ is derived using n strong transitions (i.e., we have $P(\xrightarrow{\tau})^m \xrightarrow{\mu} (\xrightarrow{\tau})^{m'} P'$ and $n = m + m' + 1$. If $s = \ell_1, \dots, \ell_n$, then we write $P \overset{s}{\Longrightarrow}$ if $P \overset{\ell_1}{\Longrightarrow} P_1 \overset{\ell_2}{\Longrightarrow} P_2 \dots P_{n-1} \overset{\ell_n}{\Longrightarrow} P_n$, for some processes $P_1, \dots, P_n$. Similarly we write $P \overset{s}{\Longrightarrow}_m$ if there are $P_1, \dots, P_n$ with $P \overset{\ell_1}{\Longrightarrow}_{m_1} P_1 \overset{\ell_2}{\Longrightarrow}_{m_2} P_2 \dots P_{n-1} \overset{\ell_n}{\Longrightarrow}_{m_n} P_n$, and $m = \Sigma_i m_i$.

DEFINITION 7.1. *Two processes P, Q of $\mathcal{L}$ are* trace equivalent, *written $P \approx_{\mathrm{tr}} Q$, if for each trace s we have $P \overset{s}{\Longrightarrow}$ iff $Q \overset{s}{\Longrightarrow}$.*

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

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

A process language is $\approx_{\mathrm{tr}}$*-safe* if $\approx_{\mathrm{tr}}$ is a congruence and $\succeq_{\mathrm{tr}}$ a precongruence.

THEOREM 7.2. *In any $\approx_{\mathrm{tr}}$-safe process language $\mathcal{L}$, a system of autonomous contractions has a unique solution for $\approx_{\mathrm{tr}}$.*

Refined forms of trace equivalence exist. For instance, *ready trace equivalence* [3, 10] combines traces with barbs. The idea of contraction and Theorem 7.2 can be adapted to ready traces by combining the treatment of traces in Definition 7.1 with the treatment of barbs in Definitions 5.3 and 6.2.

8. Non-applicability of the technique

The contraction technique may be applied to any equivalence whose observables are *finitary*, in the sense that if an observable holds then it can be reached in a finite number of transitions. Bisimilarity is in this class: the observables are the weak transitions $\overset{\mu}{\Longrightarrow}$; each use of $\overset{\mu}{\Longrightarrow}$ is finitary because obtained by composing a finite number of strong transitions ($\xrightarrow{\tau}$ and $\xrightarrow{\mu}$). Different uses of $\overset{\mu}{\Longrightarrow}$ may have different lengths, but each length is finite. The same argument holds for $\Downarrow$, the observable of contextual equivalence. In all these cases, the contraction preorder precisely arises by playing with such finite measures.

There are behavioural equivalences, however, in which the observables are not finitary. For instance, an observable may be inherently coinductive, as for observables such as infinite traces and non-termination. We illustrate the possible failure of the contraction techniques in these cases using *infinitary trace equivalence*, whereby two processes are equated if they have the same traces, including the infinite ones. It is unclear how the contraction of infinitary trace equivalence should be defined. In any case, however, 'unique solution' would fail, even for guarded and sequential contractions. As an example, consider the processes $P \overset{\mathrm{def}}{=} \Sigma_n a^n$ and $Q \overset{\mathrm{def}}{=} P + a.\, !a.\, \mathbf{0}$. These processes 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 the contraction for infinitary trace equivalence is irrelevant here, because the processes have no τ-transitions. Similar problems arise for *must equivalence*, where non-termination is observable — the same counterexample of infinitary trace equivalence applies.

9. Injecting contractions into the bisimulation game

An advantage of bisimilarity, with respect to other behavioural equivalences, is the locality of the required checks: related states only have to match each other's immediate transitions. We can inject some locality also in other equivalences by introducing the corresponding contraction into a 'bisimulation up-to' game. We illustrate this possibility with contextual equivalence, which is inductive and contextual and therefore faraway from bisimilarity and its local checks. Since we play the bisimulation game, we assume a process language $\mathcal{L}$ with an LTS semantics, on top of which contextual equivalence (and reduction semantics) is defined as in CCS.

DEFINITION 9.1 (bisimulation up-to $\succeq_{\mathrm{ctx}}$). *A relation $\mathcal{R}$ on the process language $\mathcal{L}$ is a* bisimulation up-to $\succeq_{\mathrm{ctx}}$ *if, whenever $P \mathrel{\mathcal{R}} Q$, we have:*

1. *$P \xrightarrow{\mu} P'$ implies $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \succeq_{\mathrm{ctx}} \mathcal{R} \approx_{\mathrm{ctx}} Q'$*
2. *the converse of (1) on the actions from Q.*

As in the case of ordinary bisimulation, so bisimulation up-to $\succeq_{\mathrm{ctx}}$ may be enhanced by combination with further up-to techniques. For instance, in the *bisimulation up-to $\succeq_{\mathrm{ctx}}$ and context* the requirement $P' \succeq_{\rm ctx} \mathcal{R} \approx_{\rm ctx} Q'$ on the derivatives of Definition 2.3 becomes

$$P' \succeq_{\rm ctx} \mathcal{R}^{\rm c} \approx_{\rm ctx} Q'$$

It is sufficient to analyse the most powerful technique ('up-to $\succeq_{\rm ctx}$ and context'), and the results will also hold for the weaker 'up-to $\succeq_{\rm ctx}$'. We derive soundness from that of the corresponding contraction technique.

LEMMA 9.2. *In $\mathcal{L}$, suppose $\mathcal{R}$ is a bisimulation up-to $\succeq_{\rm ctx}$ and context. Then there is a system of reduction-autonomous contractions, of the same size as $\mathcal{R}$, of which $\mathcal{R}_1$ and $\mathcal{R}_2$ are solutions for $\succeq_{\rm ctx}$.*

COROLLARY 9.3. (soundness of bisimulation up-to $\succeq_{\rm ctx}$ and context). *Suppose a relation $\mathcal{R}$ on the process language $\mathcal{L}$ is a bisimulation up-to $\succeq_{\rm ctx}$ and context. Then $\mathcal{R} \subseteq \approx_{\rm ctx}$.*

We have seen that, in the case of bisimilarity, the techniques of 'unique solution of contractions for $\approx$' and of 'bisimulation up-to contraction and context' are equivalent. For contextual equivalence, however, the former technique is more powerful. The reason is that, in the 'bisimulations up-to $\succeq_{\rm ctx}$ and context' game, laws and equalities for $\approx_{\rm ctx}$ are applied only *after* the derivatives of the processes in the pairs have been chosen. For instance, the lazy and eager servers of Section 6.1, *LS* and *ES*, cannot be a pair of a bisimulation up-to $\succeq_{\rm ctx}$ and context, for the same reason why they are not bisimilar: the challenge transition

$$ES \xrightarrow{\tau} \boldsymbol{\nu} a\,(A \mid c(z).\,(E \mid R\langle c, n, z\rangle))$$

cannot be matched by *LS*.

In some cases, however, the obstacle can be bypassed. We show this for the processes *LS* and *ES* of the server example. We relate *LS* to a contraction ES' of *ES*, obtained by abstracting the initial private communication with the auxiliary server A:

$$ES \succeq_{\rm ctx} \Sigma_n \boldsymbol{\nu} a\,(A \mid c(z).\,(E \mid R\langle c, n, z\rangle)) \overset{\rm def}{=} ES' \qquad (6)$$

Now, the singleton relation $\{(LS, ES')\}$ is a bisimulation up-to $\succeq_{\rm ctx}$ and context. The processes in the pair initially may only perform an input at c; if v is the value received in the input, then the transitions are

$$\begin{array}{lrcl} & ES' & \xrightarrow{c\langle v\rangle} & \Sigma_n \boldsymbol{\nu} a\,(A \mid E \mid R\langle c, n, v\rangle) \overset{\rm def}{=} ES'_1 \\ \text{and} & LS & = & \boldsymbol{\nu} a\,(A \mid c(z).\,a(x).\,(L \mid R\langle c, x, z\rangle)) \\ & & \xrightarrow{c\langle v\rangle} & \boldsymbol{\nu} a\,(A \mid a(x).\,(L \mid R\langle c, x, v\rangle)) \overset{\rm def}{=} LS_1 \end{array}$$

Now we have, using algebraic reasoning similar to that in previous examples with the servers and (6):

$$\begin{array}{lrcl} & ES'_1 & \sim & \Sigma_n (\boldsymbol{\nu} a\,(A \mid E) \mid R\langle c, n, v\rangle) \\ & & \succeq_{\rm ctx} & \Sigma_n (ES' \mid R\langle c, n, v\rangle) \\ \text{and} & LS_1 & \succeq_{\rm ctx} & \Sigma_n (\boldsymbol{\nu} a\,(A \mid L) \mid R\langle c, x, v\rangle) \\ & & = & \Sigma_n (LS \mid R\langle c, x, v\rangle) \end{array}$$

This is sufficient, up-to $\succeq_{\rm ctx}$ and context. Finally, having proved $ES \approx_{\rm ctx} ES'$ and $ES' \approx_{\rm ctx} LS$, we derive $ES \approx_{\rm ctx} LS$ by transitivity.

Other behavioural equivalences and their contractions can be injected into the 'bisimulation up to' game, along the lines of what done here for contextual equivalence.

10. Higher-order languages

The contraction technique may also be used in higher-order languages such as the λ-calculus and the Higher-Order π-calculus [34]. We refrain from attempting to produce a general theory of contractions for higher-order languages, comparable in power to the bisimulation proof method for these languages. We leave this for future work. Here we simply show that the transport to a higher-order setting of the most basic contraction techniques — those involving a reduction semantics and contextually-defined equivalences — can still be useful. We illustrate this on the Higher-Order π-calculus.

We consider the Higher-Order π-calculus in its simplest form, where only processes can be communicated. Below is the syntax; see Figure 2 for the reduction semantics.

$$P \quad ::= \quad \overline{a}\langle P\rangle.\,Q \;\Big|\; a(x).\,P \;\Big|\; x \;\Big|\; \boldsymbol{\nu} a\,P \;\Big|\; P \mid Q \;\Big|\; \mathbf{0}$$

We use $a, b, c, \ldots$ to range over names, and $x, y, z, \ldots$ to range over variables; we call them *language variables* (to distinguish them from the contraction or equation variables such as X, Y). An input $a(x).\,P$ binds the free occurrences of variable x in P; similarly a restriction $\boldsymbol{\nu} a\,P$ binds the free occurrences of name a in P. A term is *open* or *closed* depending on whether it may, or may not, have free language variables (in any case, it may have free names).

Systems of contractions in the Higher-Order π-calculus In the definition of barbed congruence and its contraction, the only technicality of higher-order languages that has to be taken into account is the distinction between open and closed terms. This is dealt with in the expected manner. All running terms are supposed to be closed. Thus the definitions of equivalences and preorders in earlier sections (e.g., barbed congruence and its contraction) are meant to be on closed terms. The definitions are generalised to open expressions by requiring instantiation of the language variables with all closing substitutions, i.e., substitutions that make the terms closed (using, in contextual definitions, closing contexts rather than closing substitutions would yield the same relations).

An example The example is about two ways of modeling the replication operator. We consider the equality (barbed congruence) between the terms $\overline{c}\langle A\rangle$ and $\overline{c}\langle B\rangle$, where

$$A \overset{\rm def}{=} b(y).\,\boldsymbol{\nu} a\,(M \mid \overline{a}\langle M\rangle) \quad \text{ for } M \overset{\rm def}{=} a(x).\,(y \mid x \mid \overline{a}\langle x\rangle)$$

and

$$B \overset{\rm def}{=} b(y).\,\boldsymbol{\nu} a\,(N \mid \overline{a}\langle y \mid N\rangle) \quad \text{ for } N \overset{\rm def}{=} a(x).\,(x \mid \overline{a}\langle x\rangle).$$

Terms $\overline{c}\langle A\rangle$ and $\overline{c}\langle B\rangle$ send on c processes (A and B) that can receive a process at b and then replicate this process. Indeed, if P is the process so received, assuming a does not occur free in P, in one case we obtain the term

$$A_P \overset{\rm def}{=} \boldsymbol{\nu} a\,(M\{P/y\} \mid \overline{a}\langle M\{P/y\}\rangle)$$

and in the other case

$$B_P \overset{\rm def}{=} \boldsymbol{\nu} a\,(N \mid \overline{a}\langle P \mid N\rangle)\,,$$

and then we have:

$$A_P \longrightarrow P \mid A_P \longrightarrow P \mid P \mid A_P \longrightarrow \cdots$$

$$B_P \longrightarrow P \mid B_P \longrightarrow P \mid P \mid B_P \longrightarrow \cdots$$

The internal structure of A_P and B_P is however different.

A system of contractions that proves $\overline{c}\langle A\rangle \approx^{\rm c}_{\rm bar} \overline{c}\langle B\rangle$ is the following:

$$\begin{array}{rcl} X & \succeq & \overline{c}\langle Y\rangle.\,\mathbf{0} \\ Y & \succeq & b(y).\,Z \\ Z & \succeq & \tau.\,(y \mid Z) \end{array}$$

These contractions are reduction-autonomous, and therefore have a unique solution for barbed congruence. Note that, in the first contraction, a contraction variable occurs within the initial output prefix. Thus the contraction is *not* weakly guarded. Still, the contraction is reduction-autonomous because a process that replaces the

Structural congruence the least congruence $\equiv$ such that:

- $P \mid Q \equiv Q \mid P$, $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$, $P \mid \mathbf{0} \equiv P$; $\boldsymbol{\nu} a\, \mathbf{0} \equiv \mathbf{0}$, $\boldsymbol{\nu} a\, \boldsymbol{\nu} b\, P \equiv \boldsymbol{\nu} b\, \boldsymbol{\nu} a\, P$; $(\boldsymbol{\nu} a\, P) \mid Q \equiv \boldsymbol{\nu} a\, (P \mid Q)$, if a not free in Q

The reduction relation $P \longrightarrow Q$ is the least relation such that:

$$a(x).\, R \mid \overline{a}\langle P\rangle.\, Q \longrightarrow R\{^{P}\!/\!_{x}\} \mid Q \qquad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{\boldsymbol{\nu} a\, P \longrightarrow \boldsymbol{\nu} a\, P'} \qquad \frac{P \equiv P' \longrightarrow P'' \equiv P'''}{P \longrightarrow P''''}$$

Figure 2. The reduction semantics for HOπ

variable (and that therefore represents the value emitted in the output) does not contribute to the first action. Note also that the third contraction is open — it has y as a free variable. Two solutions for $\succeq^{\mathrm{c}}_{\mathrm{bar}}$ (the barbed congruence contraction) to the above system of three contractions are, respectively:

1. $\overline{c}\langle A\rangle$, A, and $\boldsymbol{\nu} a\, (M \mid \overline{a}\langle M\rangle)$;
2. $\overline{c}\langle B\rangle$, B, and $\boldsymbol{\nu} a\, (N \mid \overline{a}\langle y \mid N\rangle)$.

The third process of each solution has y free, as its corresponding contraction. To prove that these are solutions, we need a few simple algebraic laws.

Using the bisimulation proof method, the proof of the equality between $\overline{c}\langle A\rangle$ and $\overline{c}\langle B\rangle$ is more cumbersome; with the bisimulation techniques currently available, a proof requires an infinite relation. Even in the case of environmental bisimulation, where a form of 'up-to context' is available, the relation used in [35] for the same example is infinite because, intuitively, the values emitted, A and B, have to be stored in an environment, and can then be played back at any time, possibly several times. What makes the difference is that contractions here allow us to extract a common context that incorporates the prefix for the initial action. In contrast, in 'up-to context' techniques for bisimulation, contexts are only removed from the derivatives, after firing an initial prefix.

11. Further related work

Milner's theorem about unique solution of equations stems from an axiomatisation of bisimulation on finite-state processes [21]. Indeed, in axiomatisations of behavioural equivalences [2, 19], the corresponding rule plays a key role and is called *fixed-point rule*, or *recursive specification principle*; see also [27], for trace equivalence. The possible shapes of the solutions of systems of equations, in connection with conditions on the guardedness of the equations, is studied by Baeten and Luttik [4].

Unique solution of equations has been considered in various settings, including languages, algebraic power series and pushdown automata (see the surveys [15, 23]), as well as in coalgebras (e.g., [18]). These models, however, do not have the analogous of 'internal step', around which all the theory of contractions is built. In functional languages, unique solution of equations is sometimes called 'unique fixed-point induction principle'. See for instance [31], in which the conditions resembles Milner's conditions of Theorem 2.8, and [13], which studies equations on streams advocating a condition based on the notion of 'contractive function' (the word 'contraction' here is unrelated to its use in our paper).

A tutorial on bisimulation enhancements is [26]. 'Up-to context' techniques have been formalised in a coalgebraic setting, and adapted to languages whose LTS semantics adheres to the GSOS format [5]; see for instance [6], which uses lambda-bialgebras, a generalisation of GSOS to the categorical framework.

The techniques in Section 9, transporting the bisimulation proof method and some of its enhancements onto non-coinductive equivalences, remind us of techniques for reducing non-coinductive equivalences to bisimilarity. For instance, trace equivalence on nondeterministic processes can be reduced to bisimilarity on deterministic processes, following the powerset construction for automata [12]; a similar reduction can be made for testing equivalence [8]. These results rely on transformations of transitions systems, which modify the nondeterminism and the set of states, in such a way that a given equivalence on the original systems corresponds to bisimilarity on the altered systems. In contrast, in the techniques of Section 9 the transformation of processes is performed dynamically, alongside the bisimulation game: two processes are manipulated only when necessary, i.e., when their immediate transitions would break the bisimulation game.

Some beautiful results have been obtained in CSP [28, 29] in which systems of equations have unique solutions provided their least fixed point (intuitively obtained by infinite unfolding of the equations) does not contain divergent states. In CSP the semantics has usually a denotational flavour and, most important, the reference behavioural equivalence, failure equivalence, is divergent sensitive. As mentioned in Section 8, currently we do not know how to handle divergence in the theory of contractions, as divergence is not a finitary observable. We note however that (at least in the equivalences considered in the paper) unique solution of contractions holds in cases where the infinite unfolding of the contractions would introduce divergence: e.g., the contractions of Example 3.9, as well as the contractions employed in the examples about the lazy and eager servers (where divergence may appear if, in the interaction protocol with a client, the main server is called back).

12. Conclusions and future work

In this paper we have presented operational techniques, based on the idea of contraction, for proving weak behavioural equivalences, that is, equivalences that abstract from internal moves. We have focused on concurrent languages but the techniques are not meant to be specific to concurrency. We have illustrated the techniques with bisimulation, the most natural ground of application, discussing also completeness. We have then shown that the technique of unique solution of contractions can be transported onto other equivalences, with finitary observables (e.g., contextual equivalence, barbed congruence, trace equivalence). We have also seen that the contraction preorders can be injected into the bisimulation game. In the case of bisimulation, this leads to a (minor) improvement of an existing technique, namely 'bisimulation up-to expansion and context'. The case of non-coinductive or contextual equivalences such as contextual equivalence is more interesting: we can use the bisimulation proof method (enhanced with up-to context) for reasoning on these equivalences, combined with algebraic laws for manipulating states whose immediate transitions would break the bisimulation game. Such techniques allow us, implicitly, to transfer 'up-to context' forms of reasoning, originally proposed for labeled bisimilarities and their proof method, onto equivalences that are contextual or non-coinductive.

As for the technique based on equations, so the technique based on contractions is meant to be used in combination with algebraic reasoning, on terms whose behaviour is not finite or finite-state: the recursion on the contraction variables captures the infinite behaviour of terms, and the proof that certain processes are solutions is carried out with pure algebraic reasoning.

In comparison with equations, a drawback of unique solution of contractions for an equivalence $\asymp$ is that the solutions are not $\asymp$-interchangeable: it may be that P is solution and Q is not, even though $P \asymp Q$.

The proof of completeness of the 'unique solution of contractions' method with respect to the bisimulation proof method uses the sum operator to express the possible initial actions of a process. We would like to see how completeness can be recovered in languages in which the sum operator is missing. One may consider the introduction of an operator akin to sum, to be used only for writing contractions. Also, we did not tackle completeness in equivalences other than bisimilarity.

We have related the contraction technique to bisimulation enhancements such as 'up-to expansion and context'. While powerful, these are not the only possible enhancements. It would be interesting to see whether other enhancements can be captured using contractions or similar notions.

We would like to understand on which behavioural equivalences the technique of unique solution of contractions works. We mentioned in Section 8 that it seems to work if the observables of the equivalence are finitary. More experimentation is needed to clarify this point, and formalise appropriate conditions.

In the example with a higher-order language, we have applied the most basic contraction techniques — those for contextually-defined equivalences. The use of other contraction techniques requires further investigation. Such study may shed light on the applicability of up-to context techniques to higher-order languages [14, 16, 17, 24, 35].

Our original motivation for studying contractions was to better understand 'up-to context' enhancements of the bisimulation proof method and their soundness. More broadly, the goal of the line of work reported is to improve our understanding of bisimilarity and the proof techniques for it, including the possibility of exporting the techniques onto other equivalences.

Acknowledgments

I have benefited from discussions with Luca Aceto, Jos Baeten, Filippo Bonchi, Matthew Hennessy, Bill Roscoe, David Sands, and from the comments of the anonymous referees. This work has been partially supported by the ANR project 12IS02001 'PACE', and the MIUR-PRIN project 'CINA'.

References

[1] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29:737–760, 1992.

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

[3] Jos C. M. Baeten, Jan A. Bergstra, and Jan Willem Klop. Ready-trace semantics for concrete process algebra with the priority operator. *Comput. J.*, 30(6):498–506, 1987.

[4] Jos C. M. Baeten and Bas Luttik. Unguardedness mostly means many solutions. *Theor. Comput. Sci.*, 412(28):3090–3100, 2011.

[5] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995.

[6] F. Bonchi, D. Petrisan, D. Pous, and J. Rot. Coinduction up to in a fibrational setting. Proc. CSL-LICS'14, ACM, 2014.

[7] Filippo Bonchi and Damien Pous. Checking nfa equivalence with bisimulations up to congruence. In *Proc. POPL'13*, pages 457–468. ACM, 2013.

[8] Rance Cleaveland and Matthew Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Asp. Comput.*, 5(1):1–20, 1993.

[9] R. De Nicola and R. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[10] R.J. van Glabbeek. The linear time—branching time spectrum II. In *Proc. CONCUR '93*, volume 715. Springer, 1993.

[11] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

[12] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Boston, MA, USA, 2006.

[13] G. Hutton and M. Jaskelioff. Representing Contractive Functions on Streams. Submitted, 2011.

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

[15] Michal Kunc. Simple language equations. *Bulletin of the EATCS*, 85:81–102, 2005.

[16] S.B. Lassen. Relational reasoning about contexts. In *Higher-order operational techniques in semantics*, pages 91–135. Cambridge University Press, 1998.

[17] S.B. Lassen. Bisimulation in untyped lambda calculus: Böhm trees and bisimulation up to context. *Electr. Notes Theor. Comput. Sci.*, 20:346–374, 1999.

[18] S. Milius, L. S. Moss, and D. Schwencke. Abstract GSOS rules and a modular treatment of recursive definitions. *Logical Methods in Computer Science*, 9(3), 2013.

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

[20] R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proc. 19th ICALP*, volume 623 of *LNCS*, Springer Verlag, 1992.

[21] Robin Milner. A complete axiomatisation for observational congruence of finite-state behaviors. *Inf. Comput.*, 81(2):227–247, 1989.

[22] James H. Morris. *Lambda-Calculus Models of Programming Languages*. Phd thesis MAC-TR-57, M.I.T., project MAC, Dec. 1968.

[23] Ion Petre and Arto Salomaa. Algebraic systems and pushdown automata. In *Handbook of Weighted Automata*, EATCS Series, pages 257–289. Springer, 2009.

[24] Adrien Piérard and Eijiro Sumii. Sound bisimulations for higher-order distributed process calculus. In *Proc. FOSSACS*, volume 6604 of *LNCS*, pages 123–137. Springer, 2011.

[25] Andrew Pitts. Howe's method. In *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2012.

[26] Damien Pous and Davide Sangiorgi. Enhancements of the bisimulation proof method. In *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2012.

[27] Alexander Moshe Rabinovich. A complete axiomatisation for trace congruence of finite state behaviors. In *Proc. 9th MFPS*, volume 802 of *LNCS*, pages 530–543. Springer, 1993.

[28] A. W. Roscoe. *The theory and practice of concurrency*. Prentice Hall, 1998.

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

[30] Jurriaan Rot, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Coinductive proof techniques for language equivalence. In *Proc. LATA*, volume 7810 of *LNCS*, pages 480–492. Springer, 2013.

[31] David Sands. Computing with Contexts: A simple approach. ENTCS, volume 10, Elsevier, 1998.

[32] Sangiorgi, D. and Milner, R. The problem of "Weak Bisimulation up to". In *Proc. CONCUR '92*, volume 630 of *LNCS*, Springer, 1992.

[33] D. Sangiorgi. Locality and True-concurrency in Calculi for Mobile Processes. In *Proc. TACS '94*, volume 789 of *LNCS*, Springer, 1994.

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

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