Incremental Pattern-Based Coinduction for Process Algebra and Its Isabelle Formalization

Andrei Popescu and Elsa L. Gunter

University of Illinois at Urbana-Champaign

Abstract. We present a coinductive proof system for bisimilarity in transition systems specifiable in the *de Simone* SOS format. Our coinduction is *incremental*, in that it allows building incrementally an a priori unknown bisimulation, and *pattern-based*, in that it works on equalities of process patterns (i.e., universally quantified equations of process terms containing process variables), thus taking advantage of equational reasoning in a "circular" manner, inside coinductive proof loops. The proof system has been formalized and proved sound in Isabelle/HOL.

1 Introduction

Bisimilarity is arguably the most natural equivalence on interactive processes. Assuming process transitions are labeled by (observable) actions a, processes P and P' are bisimilar iff: **(I)** whenever P can a-transit to a process Q, P' can also a-transit to some process Q' such that P' and Q' are again bisimilar; **(II)** and vice versa; **(III)** and so on, *indefinitely* (as in an infinite game).

The above informal description of the bisimilarity relation can of course be made rigorous by defining bisimilarity to be the largest *bisimulation*, i.e., the largest relation θ for which (I) and (II) hold (with "bisimilar" replaced by "in θ"). But the largest-fixpoint description loses (at least superficially) the game-theoretic flavor of the intuitive description, so we stick to the latter for a while. How would one go about proving that P and Q are bisimilar? Well, if one were allowed an infinite proof, one could try to show that each transition of P is matched by a transition of Q so that the continuations P' and Q' are (claimed to be) bisimilar (and vice versa), and then prove the bisimilarity claims about all pairs of continuations P' and Q', and so on. This way, one would build an infinite tree whose nodes contain bisimilarity claims about pairs of processes. Now assume that, while expanding the tree, one encounters a repetition of a previous claim (that appeared on an ancestor node). A reasonable "optimization" of the infinite proof would then be to stop and "seal" that node, because the bisimilarity argument for its ancestor can be repeated ad litteram. In other words, one may take the (yet unresolved!) goal of the ancestor as a hypothesis, which discharges the repetitive goal – this is the upside of trying to build an infinite proof: non-well-foundedness (i.e., circularity) works in our advantage. Assume now one finds such repetitions on all paths when building the tree. Then our bisimilarity proof is done! In terms of the fixpoint definition, we have

L. Ong (Ed.): FOSSACS 2010, LNCS 6014, pp. 109–127, 2010.

proved that the pair (P, Q) of processes located at the root are bisimilar by *coinduction*, i.e., by exhibiting a bisimulation that contains (P, Q). In terms of proof engineering however, the needed bisimulation did not appear out of nowhere, but was built incrementally from the goal, essentially by an exploration that discovered a regular pattern for an infinite proof tree. In fact, *coinductive proofs are intuitively all about discovering regular patterns.*

This paper provides formal support for this intuition. Here is an illustration of our approach, for a mini process calculus. Fix a set of actions **act** with a given silent action $\tau \in \textbf{act}$ and a map on $\textbf{act} \setminus \{\tau\}$, $a \mapsto \overline{a}$, such that $\overline{\overline{a}} = a$ for all $a \in \mathit{Act}$. The processes P are generated by the grammar: $P ::= 0 \mid a.P \mid P|Q \mid\ !P$. Thus, we have idle process, action prefix, parallel composition, and replication. "!" binds more strongly than "|". The behavior of processes is specified by the following labeled transition system:

$$\dfrac{\cdot}{a.P \overset{a}{\rightsquigarrow} P}\textsf{(PREF)} \qquad \dfrac{P_0 \overset{a}{\rightsquigarrow} Q_0}{P_0|P_1 \overset{a}{\rightsquigarrow} Q_0|P_1}\textsf{(PARL)} \qquad \dfrac{P_1 \overset{a}{\rightsquigarrow} Q_1}{P_0|P_1 \overset{a}{\rightsquigarrow} P_0|Q_1}\textsf{(PARR)}$$

$$\dfrac{P_0 \overset{a}{\rightsquigarrow} Q_0 \quad P_1 \overset{\overline{a}}{\rightsquigarrow} Q_1}{P_0|P_1 \overset{\tau}{\rightsquigarrow} Q_0|Q_1}\textsf{(PARS)} \qquad \dfrac{P \overset{a}{\rightsquigarrow} Q}{!P \overset{a}{\rightsquigarrow}\ !P|Q}\textsf{(REPL)} \qquad \dfrac{P \overset{a}{\rightsquigarrow} Q_0 \quad P \overset{\overline{a}}{\rightsquigarrow} Q_1}{!P \overset{\tau}{\rightsquigarrow}\ !P|(Q_0|Q_1)}\textsf{(REPLS)}$$

We may wish to prove in this context that parallel composition is associative and commutative and that replication absorbs self-parallel composition, i.e., that $(P_0|P_1)|P_2 = P_0|(P_1|P_2)$, $P_0|P_1 = P_1|P_0$, and $P|!P =\ !P$ for all processes P_0, P_1, P_2, P, where we write "=" for *strong bisimilarity.* In fact, assume we already proved the first two facts and are left with proving the third, $P|!P =\ !P$. For this, we first check to see if the equations we already know so far (associativity and commutativity of |) imply this new one by pure equational reasoning – no, they don't. This means we cannot discharge the goal right away, and therefore we need to perform *unfoldings* of the two terms in the goal. We unfold $P|!P$ and $!P$ until we reach hypotheses involving only process meta-variables. The upper side of Figure 1 contains all possible *derived rules* (i.e., compositions of primitive rules in the system, all the way down to non-decomposable hypotheses) that can be matched by $P|!P$ in order to infer a transition from $P|!P$. And, similarly, the lower side for the term $!P$ – in this latter case, the matched derived

$$\dfrac{P \overset{a}{\rightsquigarrow} Q}{P|!P \overset{a}{\rightsquigarrow} Q|!P}\textsf{(PARL)}\ (1) \qquad \dfrac{\dfrac{P \overset{a}{\rightsquigarrow} Q}{!P \overset{a}{\rightsquigarrow}\ !P|Q}\textsf{(REPL)}}{P|!P \overset{a}{\rightsquigarrow} P|(!P|Q)}\textsf{(PARR)}\ (2)$$

$$\dfrac{\dfrac{P \overset{a}{\rightsquigarrow} Q_0 \quad P \overset{\overline{a}}{\rightsquigarrow} Q_1}{!P \overset{\tau}{\rightsquigarrow}\ !P|(Q_0|Q_1)}\textsf{(REPLS)}}{P|!P \overset{\tau}{\rightsquigarrow} P|(!P|(Q_0|Q_1))}\textsf{(PARR)}\ (3) \qquad \dfrac{P \overset{a}{\rightsquigarrow} Q_0 \quad \dfrac{P \overset{\overline{a}}{\rightsquigarrow} Q_1}{!P \overset{\overline{a}}{\rightsquigarrow}\ !P|Q_1}\textsf{(REPL)}}{P|!P \overset{\tau}{\rightsquigarrow} Q_0|(!P|Q_1)}\textsf{(PARS)}\ (4)$$

$$\dfrac{P \overset{a}{\rightsquigarrow} Q}{!P \overset{a}{\rightsquigarrow}\ !P|Q}\textsf{(REPL)}\ (5) \qquad \dfrac{P \overset{a}{\rightsquigarrow} Q_0 \quad P \overset{\overline{a}}{\rightsquigarrow} Q_1}{!P \overset{\tau}{\rightsquigarrow}\ !P|(Q_0|Q_1)}\textsf{(REPLS)}\ (6)$$

Fig. 1. The matching derived rules for $P|!P$ and $!P$

rules coincide with the matched primitive rules. To see how the derived rules are obtained, the figure shows whole *derivation trees*, but we only care about the leaves and the roots of these trees.

Next, we try to pair these derived rules (upper versus lower), by the accordance of their hypotheses and their transition labels. The only valid pairing possibilities are: (1) with (5), (2) with (5), (3) with (6), and (4) with (6). The targets of the conclusions of the rules in these pairs yield four new goals: **(i)** $Q|!P = !P|Q$; **(ii)** $P|(!P|Q) = !P|Q$; **(iii)** $P|(!P|(Q_0|Q_1)) = !P|(Q_0|Q_1)$; **(iv)** $Q_0|(!P|Q_1) = !P|(Q_0|Q_1)$. The original goal, $P|!P = !P$, is replaced by the above four goals, *and is also henceforth taken as a hypothesis*. Notice that our goals are generic, i.e., universally quantified over the occurring process meta-variables, P, Q, Q_0, Q_1. Now, *equational reasoning* (by standard equational rules, *including substitution*) with hypothesis $P|!P = !P$ together with the already known lemmas $(P_0|P_1)|P_2 = P_0|(P_1|P_2)$ and $P_0|P_1 = P_1|P_0$ is easily seen to discharge each of the remaining four goals, and the proof is done.

Why is this proof valid, i.e., why does it represent a proof of the fact that, for all process terms P, $P|!P$ and $!P$ are bisimilar? The rigorous justification for this is the topic of this paper. But the short answer has to do with our previous discussion on discovering patterns: the above is really a proof by coinduction (on universally quantified equalities of terms up to equational closure), which *builds incrementally* the relation representing the coinductive argument. Notice the appearance of *circular reasoning*: a goal that cannot be locally discharged is expanded according to the SOS definition transition relation and *becomes a hypothesis*. In this particular example, the proof is finished after only one expansion, but the process of expanding the goals with taking them as hypotheses may in principle continue, obtaining arbitrarily large proof trees.

We show that deductions such as the above are sound for a wide class of process algebras – those specifiable by SOS rules in the de Simone format [11]. Our results have been given a formalization in Isabelle/HOL [3], which was desirable for two reasons: first, the very technical constructions (especially in Sec. 4) and arguments (in both Secs. 3 and 4) were asking for a means to be absolutely sure of their correctness; second, the formalization has the potential of leading to an implementation of a coinductive tool. Here is the structure of this paper. The rest of this section establishes some notation. Sec. 2 discusses our representation of the de Simone format. Secs. 3 and 4 contain our original theoretic contribution: incremental proof systems for bisimilarity – Sec. 3 for standard bisimilarity, Sec. 4 for universally quantified bisimilarity equations. Sec. 5 discusses related and future work. More details on our Isabelle scripts and on various other technical topics, as well as more examples, can be found in the technical report [28], to which we occasionally refer from this paper. [28] is an identical copy of this paper, except that it has an appendix with more details. The Isabelle scripts can be found at http://hdl.handle.net/2142/14857 in both html-browsable and pdf formats (App. C in [28] has details about the scripts.)

Conventions and notations. By "Isabelle", here we mean "Isabelle/HOL". We present our work in the usual mathematical language, but partly employ the *Isabelle dialect* of this language in order to allow the interested reader to easily relate this paper with our Isabelle formal proofs. (We believe that this choice does not decrease readability, since the Isabelle notation is very close to standard mathematical notation and also occasionally allows for clear and concise formulations, as, e.g., with datatypes and records. A priori familiarity with the Isabelle dialect is *not* required from the reader.)

Isabelle distinguishes between a *type* and a *set*, but the set-theoretical-oriented reader is free to ignore this distinction; as a matter of syntax though, membership to a type is denoted by "::" and membership to a set by "$\in$". **nat** is the type of naturals. Given types α and β, $\alpha \times \beta$ is their product type, $\alpha \Rightarrow \beta$ the type of functions between α and β, α $\underline{\textbf{list}}$ the type of lists of items in α, and α $\underline{\textbf{set}}$ the type of sets of items in α.[1] fst and snd are the two projections from $\alpha \times \beta$. $[]$ is the empty list and $[a_0, \ldots, a_{n-1}]$ the list consisting of the n indicated items; given a list L and $i < \mathsf{length}\, L$, $L!i$ is the $(i+1)$-th element in L (thus, the first element is $L!0$). The operator $\mathsf{set} : \alpha\ \underline{\textbf{list}} \Rightarrow \alpha\ \underline{\textbf{set}}$ gives the set of the items appearing in a list. A list L is said to be *nonrepetitive* if $L!i \neq L!j$ for all $i, j < \mathsf{length}\ L$ with $i \neq j$. For readability, we consistently use: sans serif fonts for constants, such as length and set; boldface for types, such as **nat**; underlined boldface for type constructors, such as $\underline{\textbf{list}}$ and $\underline{\textbf{set}}$.

2 Syntax and Operational Semantics of Processes

Process variables, terms and substitution. We fix the following types: **param**, of *parameters*, ranged over by p; **opsym**, of *operation symbols* (*opsyms* for short), ranged over by f, g; **var**, of *(process) variables*, ranged over by X, Y, Z – this latter type is assumed to be infinite. The type **term**, of *(process) terms*, ranged over by P, Q, R, T, S, U, V, is defined as an Isabelle datatype (i.e., initial algebra): DATATYPE **term** $=$ Var **var** $\mid$ Op **opsym** (**param** $\underline{\textbf{list}}$) (**term** $\underline{\textbf{list}}$).

Thus, a term can have any opsym at the top, applied to any list of parameters and any list of terms (of any length), without being subject to further well-formedness conditions. Hence an opsym f does *not* have an a priori associated numeric rank (m, n) (indicating that f takes m parameters and n terms). Rather, we allow in **term** the whole pool of all possible terms under all possible rankings of the operation symbols. This looseness w.r.t. terms is admittedly a formalization shortcut (fitting nicely the Isabelle simply-typed framework), but is completely unproblematic for the concepts and results of this paper: while an SOS specification of a transition system will of course employ only certain (possibly overloaded) ranks for the opsyms, the unused ranks will be harmless, since they will not affect transitions or bisimilarity.

[1] Note the use of postfix notation for type constructors – this is not standard mathematically, but is intuitive, as it matches natural language: while an element of **int** is an integer, an element of **int** $\underline{\textbf{list}}$ is an integer list.

σ and τ will range over **var** ⇒ **term**. We consider the operators:
- vars :: **term** ⇒ **var** **<u>set</u>**, giving the set of variables occurring in a term.
- $\_[\_]$ :: **term** × (**var** ⇒ **term**) ⇒ **term**, such that $T[\sigma]$ is the term obtained from T by substituting all its variables X by $\sigma\,X$.

Next we represent the meta-SOS notion of a *transition-system specification* [15,26]. Given any type α, the type α **<u>ftrans</u>**, of formal α-transitions, consists of pairs, written $k \rightsquigarrow l$, with $k, l :: \alpha$, where k is called the *source* and l the *target.* We fix a type **act**, of *actions*, ranged over by a, b.

Rules syntax. The type **rule**, of *(SOS-)rules*, ranged over by rl, is defined to be the following record type (i.e., a product with named projections): Record **rule** =

hyps :: (**var** **<u>ftrans</u>**) **<u>list</u>** (read "hypotheses")
cnc :: **term** **<u>ftrans</u>** (read "conclusion")
side :: (**nat** ⇒ **act**) ⇒ **act** ⇒ **bool** (read "side-condition")

The hypotheses and the conclusions of our rules are therefore formal transitions between variables, and between terms, respectively. I.e., for any rule rl:
- $\mathsf{hyps}\ rl$ has the form $[XX_0 \rightsquigarrow Y_0, \ldots, XX_{n-1} \rightsquigarrow Y_{n-1}]$, with XX_j, Y_j variables;
- $\mathsf{cnc}\ rl$ has the form $S \rightsquigarrow T$, with S and T terms.

One can visualize rl as

$$\frac{XX_0 \rightsquigarrow Y_0,\ \ldots\ , XX_{n-1} \rightsquigarrow Y_{n-1}}{S \rightsquigarrow T}\ [\lambda\, as, b.\ \mathsf{side}\ rl\ as\ b]$$

where as :: **nat** ⇒ **act** and b :: **act**. Actually, we think of rl as follows:

$$\frac{XX_0 \overset{as\ 0}{\rightsquigarrow} Y_0\ ,\ \ldots\ , XX_{n-1} \overset{as\ (n-1)}{\rightsquigarrow} Y_{n-1}}{S \overset{b}{\rightsquigarrow} T}\ [\mathsf{side}\ rl\ as\ b]$$

Note however that the side condition $\mathsf{side}\ rl$ is (for now) allowed to take into consideration the whole function as, and not only its first n values $as\ 0, \ldots, as\ (n-1)$, as one would expect – this is corrected below by "saneness".

Given a rule rl with $\mathsf{hyps}\ rl$ and $\mathsf{cnc}\ rl$ as above, we write: $\mathsf{theXXs}\ rl$, for the variable list $[XX_0, \ldots, XX_{n-1}]$; $\mathsf{theYs}\ rl$, for the variable list $[Y_0, \ldots, Y_{n-1}]$; $\mathsf{theS}\ rl$, for the term S; $\mathsf{theT}\ rl$, for the term T.

A rule rl is said to be *sane* if the following hold:

-**(1)** $\mathsf{theYs}\ rl$ is nonrepetitive;
-**(2)** $\mathsf{set}(\mathsf{theXXs}\ rl) \subseteq \mathsf{vars}(\mathsf{theS}\ rl)$;
-**(3)** $\mathsf{vars}(\mathsf{theS}\ rl) \cap \mathsf{set}(\mathsf{theYs}\ rl) = \emptyset$;
-**(4)** $\mathsf{vars}(\mathsf{theT}\ rl) \subseteq \mathsf{vars}(\mathsf{theS}\ rl) \cup \mathsf{set}(\mathsf{theYs}\ rl)$;
-**(5)** $\forall as, as'.\ (\forall i < \mathsf{length}(\mathsf{theYs}\ rl).\ as\ i = as'\ i) \longrightarrow \mathsf{side}\ rl\ as = \mathsf{side}\ rl\ as'$.

A rule rl is said to be *amenable* if $\mathsf{theS}\ rl$ has the form $\mathsf{Op}\ f\ ps\ [\mathsf{Var}\ X_0, \ldots, \mathsf{Var}\ X_{m-1}]$, where f is an opsym, ps a list of parameters, and $[X_0, \ldots, X_{m-1}]$ a *nonrepetitive* list of variables. Given an amenable rule rl as above, we write $\mathsf{thef}\ rl$ for f, $\mathsf{theps}\ rl$ for ps, and $\mathsf{theXs}\ rl$ for $[X_0, \ldots, X_{m-1}]$.

Saneness expresses a natural property for well-behaved SOS rules: Think of a term S as a generic composite process, built from its unspecified components (its variables) by means of opsyms. Then a sane rule is one that describes the behavior of the composite S in terms of the behavior of (some of) its components: condition (2) says that indeed the hypotheses refer to the components, (1) and (3) that the hypotheses only assume that some components transit "somewhere" (without any further information), (4) that the resulted continuation of the composite depends only on the components and their continuations, and (5) that the side-condition may only depend on the action labels of the hypotheses and of the conclusion. In addition, amenability asks that the composite process S be obtained by a primitive operation f applied to unspecified components. The conjunction of saneness and amenability is precisely the de Simone format requirement [11], hence we call a rule *de Simone* if it is sane and amenable.

Running example. We show what the example in the introduction becomes under our representation. Assume that **act** is an unspecified type with constants $\overline{} :: \mathbf{act} \Rightarrow \mathbf{act}$ and $\tau :: \mathbf{act}$ such that $\overline{\overline{a}} = a$ for all $a \neq \tau$. Define the relation $\mathsf{sync} :: \mathbf{act} \Rightarrow \mathbf{act} \Rightarrow \mathbf{act} \Rightarrow \mathbf{bool}$ by $\mathsf{sync}\ a\ b\ c = (a \neq \tau \wedge b \neq \tau \wedge \overline{a} = b \wedge c = \tau)$. We take **opsym** to be a three-element datatype $\mathsf{Pref} \mid \mathsf{Par} \mid \mathsf{Repl}$ and **param** to be **act**. For readability, in our running example (including throughout the future continuations of this example), for all $X :: \mathbf{var}$, $S, T :: \mathbf{term}$ and $a :: \mathbf{act}$, we use the following abbreviations: X for $\mathsf{Var}\ X$; $a.S$ for $\mathsf{Op}\ \mathsf{Pref}\ [a]\ [S]$; $S \,|\, T$ for $\mathsf{Op}\ \mathsf{Par}\ []\ [T, S]$; $!\,S$ for $\mathsf{Op}\ \mathsf{Repl}\ []\ [S]$.

Rls consists of the rules $\{\mathsf{PREF}_a .\ a :: \mathbf{act}\} \cup \{\mathsf{PARL}, \mathsf{PARR}, \mathsf{PARS}, \mathsf{REPL}, \mathsf{REPLS}\}$ listed below, where X, Y, X_0, X_1, Y_0, Y_1 are fixed *distinct* variables.

$$\frac{\cdot}{a.X \overset{b}{\rightsquigarrow} X}\,(\mathsf{PREF}_a)\,[a = b] \qquad \frac{X_0 \overset{as\,0}{\rightsquigarrow} Y_0}{X_0 \mid X_1 \overset{b}{\rightsquigarrow} Y_0 \mid X_1}\,(\mathsf{PARL})\,[as\,0 = b]$$

$$\frac{X_0 \overset{as\,0}{\rightsquigarrow} Y_0}{X_1 \mid X_0 \overset{b}{\rightsquigarrow} X_1 \mid Y_0}\,(\mathsf{PARR})\,[as\,0 = b] \qquad \frac{X_0 \overset{as\,0}{\rightsquigarrow} Y_0 \qquad X_1 \overset{as\,1}{\rightsquigarrow} Y_1}{X_0 \mid X_1 \overset{b}{\rightsquigarrow} Y_0 \mid Y_1}\,(\mathsf{PARS})\,[\mathsf{sync}\,(as\,0)\,(as\,1)\,b]$$

$$\frac{X \overset{as\,0}{\rightsquigarrow} Y}{!X \overset{b}{\rightsquigarrow} !X \mid Y}\,(\mathsf{REPL})\,[as\,0 = b] \qquad \frac{X \overset{as\,0}{\rightsquigarrow} Y_0 \qquad X \overset{as\,1}{\rightsquigarrow} Y_1}{!X \overset{b}{\rightsquigarrow} !X \mid (Y_0 \mid Y_1)}\,(\mathsf{REPLS})\,[\mathsf{sync}\,(as\,0)\,(as\,1)\,b]$$

For listing the rules, we employed the previously discussed visual representation. E.g., the formal description of PARS is $(\!|\ \mathsf{hyps} = [X_0 \rightsquigarrow Y_0,\ X_1 \rightsquigarrow Y_1];\ \mathsf{cnc} = (X_0 \mid X_1\ \rightsquigarrow\ Y_0 \mid Y_1);\ \mathsf{side} = (\lambda\ as, b.\ \mathsf{sync}\,(as\,0)\,(as\,1)\,b)\ |\!)$. All the rules in this example are easily seen to be de Simone.

Rules semantics. We fix *Rls*, a set of *de Simone* rules. The one-step transition relation induced by *Rls* on terms is a (curried) ternary relation $\mathsf{step} :: \mathbf{term} \Rightarrow \mathbf{act} \Rightarrow \mathbf{term} \Rightarrow \mathbf{bool}$, where we write $P \overset{a}{\rightsquigarrow} Q$ instead of $\mathsf{step}\ P\ a\ Q$, defined inductively by the following clause:

- if $rl \in Rls$, $\sigma((\mathsf{theXXs}\ rl)!j) \overset{as\ j}{\rightsquigarrow} \sigma((\mathsf{theYs}\ rl)!j)$ for all $j < \mathsf{length}(\mathsf{theYs}\ rl)$, and $\mathsf{side}\ rl\ as\ b$ holds, then $(\mathsf{theS}\ rl)[\sigma] \overset{b}{\rightsquigarrow} (\mathsf{theT}\ rl)[\sigma]$

(where $\sigma :: \mathbf{var} \Rightarrow \mathbf{term}$, $as :: \mathbf{nat} \Rightarrow \mathbf{act}$, and $b :: \mathbf{act}$).

The above definition is the expected one: each (generic) rule in *Rls* yields, for each substitution of the variables in the rule and for each choice of the actions fulfilling the side-condition, an inference of the instance of the rule's conclusion from the instances of the rule's hypotheses.

Bisimilarity. We write **rel** for (**term** × **term**) <u>**set**</u>, the type of relations between terms, ranged over by θ, η, ξ. The (monotonic) *retract* operator $\mathsf{Retr} :: \mathbf{rel} \Rightarrow \mathbf{rel}$, named so because it maps each θ to a relation retracted (w.r.t. transitions) back from θ, is defined by: $\mathsf{Retr}\ \theta = \{(P, Q).\ (\forall a, P'.\ P \overset{a}{\rightsquigarrow} P' \longrightarrow (\exists Q'.\ (P', Q') \in \theta \wedge Q \overset{a}{\rightsquigarrow} Q'\ )) \wedge (\forall a, Q'.\ Q \overset{a}{\rightsquigarrow} Q' \longrightarrow (\exists P'.\ (P', Q') \in \theta \wedge P \overset{a}{\rightsquigarrow} P'\ ))\}$. The *bisimilarity* relation, $\mathsf{bis} :: \mathbf{rel}$, is the *greatest fixed point* of Retr.

Notice that we defined bisimilarity for *open* terms (i.e., terms possibly containing variables), while often in the literature both transition and bisimilarity are defined for *closed* terms only (with step and Retr defined by the same conditions as above, but acting on closed terms and relations on closed terms, respectively). However, for the de Simone format of our rules (as well as for more general formats, e.g., well-founded pure tyft [15]), transition does *not* bring any variables (in the sense that, if $P \overset{a}{\rightsquigarrow} P'$, then the free variables of P are among those of P') implying that two closed terms are bisimilar according to our definition iff they are bisimilar according to the aforementioned "closed" version.

Because of the particular format of the rules, bis is a congruence on terms. This is in fact true for rule formats more expressive than the one considered here [6,15,32]. However, we shall need to exploit a stronger property specific to the de Simone format, namely: whenever θ is a congruence, it follows that $\theta \cap (\mathsf{Retr}\ \theta)$ is also a congruence. Let, for any relation θ, $\mathsf{congCl}\ \theta$ be its congruence closure. From the above, we infer a powerful "up to" coinduction rule (that is, up to bisimilarity and up to arbitrary contexts), due to de Simone [11] and Sangiorgi [34], improving on traditional coinduction:

Theorem 1. *For all* $\theta :: \mathbf{rel}$*, if* $\theta \subseteq \mathsf{Retr}(\mathsf{congCl}(\theta \cup \mathsf{bis}))$*, then* $\theta \subseteq \mathsf{bis}$*.*

3 The Raw Coinductive Proof System

We now present the core of our original theoretical contribution: defining an incrementally-coinductive proof system for bisimilarity and proving it sound. We define the *raw deduction* relation $\Vdash :: \mathbf{rel} \Rightarrow \mathbf{rel} \Rightarrow \mathbf{bool}$ (with infix notation) inductively by the clauses:

$$\boxed{\frac{\cdot}{\theta \Vdash \theta'}\mathsf{(Ax)}\,[\theta' \subseteq \mathsf{congCl}(\theta \cup \mathsf{bis})]} \boxed{\frac{\forall \theta' \in \Theta.\ \theta \Vdash \theta'}{\theta \Vdash \bigcup \Theta}\mathsf{(Split)}\,[\Theta \neq \emptyset]} \boxed{\frac{\theta' \cup \theta \Vdash \theta''}{\theta \Vdash \theta'}\mathsf{(Coind)}\,[\theta' \subseteq \mathsf{Retr}\ \theta'']}$$

$\theta \Vdash \theta'$ is eventually intended to mean: "θ implies θ' modulo bisimilarity and congruence closure". Here is the intuitive reading of the rules (thinking of them as being applied backwards for expanding or discharging goals). (Ax) allows to deduce θ' from θ right away. (Split) allows for splitting the goal according to a chosen partition of its conclusion. (Coind) is the interesting rule, and is the actual engine of the proof system. To get an intuitive grasp of this rule, let us first assume that $\theta = \emptyset$ (i.e., that θ is empty). Then the goal is to show θ' included in

$\mathsf{congCl}\ \mathsf{bis}$, i.e., in bis. For this, it would suffice that $\theta' \subseteq \mathsf{Retr}(\theta')$; alternatively, we may "defer" the goal by coming up with an "interpolant" θ'' such that $\theta' \subseteq \mathsf{Retr}(\theta'')$ and θ' implies θ'' modulo bisimilarity and congruence. (As we shall see in the next section, working symbolically with open terms provides natural interpolant candidates.) In case $\theta \neq \emptyset$, θ should be thought of *temporally* as the collection of auxiliary facts gathered from previous coinductive expansions.

Note that, for the aforementioned intention of the proof system, (Coind) is not sound *by itself*: regarded as applied backwards to a goal, it moves the conclusion θ' to the hypotheses, creating a circularity. In other words, of course it is not true that the conjunction of $\theta'' \subseteq \mathsf{congCl}(\theta' \cup \theta \cup \mathsf{bis})$ and $\theta' \subseteq \mathsf{Retr}\ \theta''$ implies $\theta' \subseteq \mathsf{congCl}(\theta \cup \mathsf{bis})$ for all $\theta, \theta', \theta''$. Yet, *the proof system as a whole* is sound in the following sense:

Theorem 2. *If* $\emptyset \Vdash \theta$*, then* $\theta \subseteq \mathsf{bis}$*.*

In the remainder of this section, we outline the proof of this theorem.

(I) In order to gain more control on the proof system, we *objectify* it in a standard fashion, by considering proofs (i.e., proof trees) explicitly, at the object level (as opposed to merely implicitly as they appear in the inductive definition of $\Vdash$). For this, we pick a sufficiently large type **index**, ranged over by i, and define the type **prf**, of *proof trees*, ranged over by *Pf*, with constructors mirroring the clauses in the definition of $\Vdash$:

$$\textsc{Datatype}\ \mathbf{prf}\ =\ \mathsf{Ax}\ \mathbf{rel}\ \mathbf{rel}\ |\ \mathsf{Split}\ (\mathbf{index} \Rightarrow \mathbf{prf})\ \mathbf{rel}\ \mathbf{rel}\ |\ \mathsf{Coind}\ \mathbf{prf}\ \mathbf{rel}\ \mathbf{rel}$$

We let *Pfs* range over **index** ⇒ **prf**. The pair of relations that a proof tree *Pf* "proves", which is (θ, θ') when *Pf* has the one of the forms $\mathsf{Ax}\ \theta\ \theta'$, $\mathsf{Split}\ \mathit{Pfs}\ \theta\ \theta'$, or $\mathsf{Coind}\ \mathit{Pf}\ \theta\ \theta'$, is denoted by $\mathsf{proves}\ \mathit{Pf}$. The conclusion-hypothesis dependencies and the side-conditions of the clauses defining $\Vdash$ are captured by the predicate $\mathsf{correct} :: \mathbf{prf} \Rightarrow \mathbf{bool}$, defined recursively as expected:

- $\mathsf{correct}\,(\mathsf{Ax}\ \theta\ \theta') = (\theta' \subseteq \mathsf{congCl}(\theta \cup \mathsf{bis}))$;
- $\mathsf{correct}\,(\mathsf{Split}\ \mathit{Pfs}\ \theta\ \theta') =$
$((\forall i.\ \mathsf{correct}(\mathit{Pfs}\ i) \wedge \mathsf{fst}(\mathsf{proves}(\mathit{Pfs}\ i)) = \theta) \wedge \bigcup i.\ \mathsf{snd}(\mathsf{proves}(\mathit{Pfs}\ i)) = \theta')$;
- $\mathsf{correct}\,(\mathsf{Coind}\ \mathit{Pf}\ \theta\ \theta') =$
$(\mathsf{correct}\,\mathit{Pf} \wedge \mathsf{fst}(\mathsf{proves}\,\mathit{Pf}) = \theta' \cup \theta \wedge \theta' \subseteq \mathsf{Retr}(\mathsf{snd}(\mathsf{proves}\,\mathit{Pf})))$.

It is immediate that $\theta \Vdash \theta'$ holds iff $\exists \mathit{Pf}.\ \mathsf{correct}(\mathit{Pf}) \wedge \mathsf{proves}(\mathit{Pf}) = (\theta, \theta')$.

(II) Thus, it suffices to show that $\theta \subseteq \mathsf{bis}$ whenever there exists a correct proof tree *Pf* such that $\mathsf{proves}(\mathit{Pf}) = (\theta, \theta')$. For showing the latter, we introduce a couple of auxiliary concepts. Given *Pf*, a *label* in *Pf* is a pair (θ, θ') "appearing" in *Pf* – formally, we define $\mathsf{labels} :: \mathbf{prf} \Rightarrow (\mathbf{rel} \times \mathbf{rel})\ \underline{\mathbf{set}}$ by:

- $\mathsf{labels}\,(\mathsf{Ax}\ \theta\ \theta') = \{(\theta, \theta')\}$;
- $\mathsf{labels}\,(\mathsf{Split}\ \mathit{Pfs}\ \theta\ \theta') = \{(\theta, \theta')\} \cup \bigcup i.\ \mathsf{labels}(\mathit{Pfs}\ i)$;
- $\mathsf{labels}\,(\mathsf{Coind}\ \mathit{Pf}\ \theta\ \theta') = \{(\theta, \theta')\} \cup \mathsf{labels}\,\mathit{Pf}$.

We let $\mathsf{Left}\,\mathit{Pf}$ denote the union of the lefthand sides of all labels in *Pf*, and $\mathsf{Right}\,\mathit{Pf}$ the union of the righthand sides of all labels in *Pf*.

Lemma 1. *If Pf is correct, then* $\mathsf{Right}\ \mathit{Pf} \subseteq \mathsf{congCl}((\mathsf{Left}\ \mathit{Pf}) \cup \mathsf{bis})$*.*

Lemma 2. *If Pf is correct and* $\mathsf{fst}(\mathsf{proves}\ \mathit{Pf}) \subseteq \mathsf{Retr}(\mathsf{Right}\ \mathit{Pf})$*, then* $\mathsf{Left}\ \mathit{Pf} \subseteq \mathsf{Retr}(\mathsf{Right}\ \mathit{Pf})$*.*

Lemma 1 follows by an easy induction on proof trees. By contrast, Lemma 2 requires some elaboration – before getting into that, let us show how the two lemmas imply our desired fact. Assume that *Pf* is correct and $\mathsf{proves}\ Pf = (\emptyset, \theta)$. Then the hypotheses of both lemmas are satisfied by *Pf*, and therefore (since also Retr is monotonic) $\mathsf{Left}\ Pf \subseteq \mathsf{Retr}(\mathsf{Right}\ Pf) \subseteq \mathsf{Retr}(\mathsf{congCl}((\mathsf{Left}\ Pf) \cup \mathsf{bis}))$, implying, by Theorem 1, $\mathsf{Left}\ Pf \subseteq \mathsf{bis}$. With Lemma 1, we obtain $\mathsf{Right}\ Pf \subseteq \mathsf{congCl}(\mathsf{bis})$, which means (given that bis is a congruence) $\mathsf{Right}\ Pf \subseteq \mathsf{bis}$. And since $\theta \subseteq \mathsf{Right}\ Pf$, we obtain $\theta \subseteq \mathsf{bis}$, as desired.

It remains to prove Lemma 2. This lemma states a property of proof trees that depends on a hypothesis concerning their roots (i.e., the pair (θ, θ') that they "prove"). The task of finding a strengthening of that hypothesis so that a direct proof by structural induction goes through seems rather difficult, if not impossible. We instead take the roundabout route of identifying an invariant satisfied on backwards paths in the proof trees whose roots satisfy our hypothesis. First, we define the notion of a *path* (independently of proof trees): a list $[(\theta_0, \theta'_0), \ldots, (\theta_{m-1}, \theta'_{m-1})]$ is called a *path* if the following is true for all $n < m-1$: either $\theta_{n+1} = \theta_n$, or $\theta_{n+1} \subseteq \mathsf{Retr}(\theta'_{n+1}) \cup \theta_n$. Then one can verify the following:
-(a) Fix $\xi :: \mathbf{rel}$. If $[(\theta_0, \theta'_0), \ldots, (\theta_{m-1}, \theta'_{m-1})]$ is a path, $\theta_0 \subseteq \mathsf{Retr}\ \xi$ and $\forall n < m.\, \theta'_n \subseteq \xi$, then $\forall n < m.\, \theta_n \subseteq \mathsf{Retr}\ \xi$. (By easy induction on n.)
-(b) If *Pf* is correct, $\mathsf{proves}(Pf) = (\theta, \theta')$, and (η, η') is a label in *Pf*, then there exists a path $[(\theta_0, \theta'_0), \ldots, (\theta_{m-1}, \theta'_{m-1})]$ consisting of labels in *Pf* (i.e., such that (θ_n, θ'_n) are labels in *Pf* for all $n < m$) and connecting (θ, θ') with (η, η') (i.e., such that $(\theta_0, \theta'_0) = (\theta, \theta')$ and $(\theta_{m-1}, \theta'_{m-1}) = (\eta, \eta')$). (By induction on *Pf*.)

With these preparations, we can prove Lemma 2: Assume $\mathsf{proves}(Pf) = (\theta, \theta')$ and $\theta \subseteq \mathsf{Retr}(\mathsf{Right}\ Pf)$. Fix a label (η, η') in *Pf*. According to (b), there exists a path connecting (θ, θ') with (η, η') and going through labels in *Pf* only. Then the hypotheses of (a) are satisfied by the aforementioned path and $\xi = \mathsf{Right}\ Pf$, and therefore all the lefthand sides of the pairs in this path are included in $\mathsf{Retr}(\mathsf{Right}\ Pf)$. In particular, $\eta \subseteq \mathsf{Retr}(\mathsf{Right}\ Pf)$. Since the choice of the label (η, η') was arbitrary, it follows that $\mathsf{Left}\ Pf \subseteq \mathsf{Retr}(\mathsf{Right}\ Pf)$, as desired.

Remarks. (1) The soundness of $\Vdash$ was established not locally (rule-wise), as is customary in soundness results, but globally, by analyzing entire proof trees. What the potential backwards applications of the clause (Coind) do is to *improve the candidate relation for the coinductive argument.* In the end, as shown by the proof of Theorem 2, the (successful) relation is synthesized by putting together the righthand sides of all labels in the proof tree.

(2) The proof system represented by $\Vdash$ is not a typical syntactic system, but contains semantic intrusions – in effect, the system is complete already by its axiom (Ax), which allows for an instantaneous "oracle proof" that the considered relation is included in bisimilarity. But of course, the realistic employment of this system will appeal to such instantaneous proofs only through the available (already proved) lemmas. (Thus, the purpose of including bis in the side-condition of (Ax) was not to ensure completeness (in such a trivial manner), but to allow the usage of previously known facts about bisimilarity.) A more syntactic and syntax-driven system for terms (also featuring oracles though, for the same reason as this one) will be presented in the next section.

4 Deduction of Universally Quantified Bisimilarity Equations

Next we introduce a deduction system for term equalities, where, as before, we interpret equality as bisimilarity, but now we interpret the occurring variables as being *universally quantified over the domain of terms.*

Universal bisimilarity, $\mathsf{ubis} :: \mathbf{rel}$, is defined as follows: $(U, U') \in \mathsf{ubis}$ iff $(U[\tau], U'[\tau]) \in \mathsf{bis}$ for all substitutions $\tau :: \mathbf{var} \Rightarrow \mathbf{term}$. Thus, e.g., given distinct variables X and Y and an opsym f, $(\mathsf{Op}\ f\ []\ [\mathsf{Var}\ X, \mathsf{Var}\ Y], \mathsf{Op}\ f\ []\ [\mathsf{Var}\ Y, \mathsf{Var}\ X])$ $\in \mathsf{ubis}$ is equivalent to $\forall U, V :: \mathbf{term}.\ (\mathsf{Op}\ f\ []\ [U, V], \mathsf{Op}\ f\ []\ [V, U]) \in \mathsf{bis}$.

Matched derived rules. Derived rules appear by composing primitive rules (i.e., the de Simone rules in *Rls*) within maximal composition chains. I.e., they come from considering, in the SOS system, derivation trees that are completely backwards-saturated (in that their leaves involve only variables as sources and targets) and then forgetting the intermediate steps in these trees. A derived rule may not be amenable (hence not de Simone), but will always be sane. We shall let *drl* denote derived rules, keeping the symbol *rl* for primitive rules.

We are interested in constructing all derived rules that are matched by a given term U in such a way that U becomes the source of the conclusion of the derived rule; in doing so, we also care about avoiding any overlap between the freshly generated variables (required to build the rules) and the variables of another given term V (that we later wish to prove universally bisimilar with U). We thus introduce the operator $\mathsf{mdr} :: \mathbf{term} \Rightarrow \mathbf{term} \Rightarrow \mathbf{rule}\ \underline{\mathbf{set}}$, read "matched derived rules", such that, given $U, V :: \mathbf{term}$, $\mathsf{mdr}\ V\ U$ is the set of all the derived rules with U as the source of their conclusion and with "the Ys" fresh for V. We write $\mathsf{mdr}_V\, U$ instead of $\mathsf{mdr}\ V\ U$.

The definition of mdr is both intuitive and standard (and was already sketched in the pioneering paper [11]), but its formalities are very technical, due to the need to avoid name overlapping and compose side-conditions. Here, we count on its understanding by examples and by its abstract properties, but App. A in [28] gives the general definition. (In [6,4], where what we call "matched derived rules" are called "ruloids", mdr is not even defined, but rather the existence of such an operator satisfying suitable properties (essentially the same with what we call below soundness and completeness of the matched derived rules) is proved.)

Running example (continued). We again assume that all the variables X, Y etc. that we refer to below are *fixed distinct variables.*
- $\mathsf{mdr}_{!X}(X \mid !X)$, the set of derived rules matched by $X \mid !X$ and with "the Ys" avoiding the variables of $!X$, consists of $\{\mathsf{DRL}_1, \mathsf{DRL}_2, \mathsf{DRL}_3, \mathsf{DRL}_4\}$ (see below);
- $\mathsf{mdr}_{X \mid !X}(!X)$, the set of derived rules matched by $!X$ and with "the Ys" avoiding the variables of $X \mid !X$, consists of $\{\mathsf{DRL}_5, \mathsf{DRL}_6\}$ (given below).

$$\dfrac{X \overset{as\,0}{\rightsquigarrow} Y}{X \mid !X \overset{b}{\rightsquigarrow} Y \mid !X}\,(\mathsf{DRL}_1)\,[as\ 0 = b] \qquad \dfrac{X \overset{as\,0}{\rightsquigarrow} Y}{X \mid !X \overset{b}{\rightsquigarrow} X \mid (!X \mid Y)}\,(\mathsf{DRL}_2)\,[\exists c.\ as\ 0 = c \wedge c = b]$$

$$\dfrac{X \overset{as\,0}{\rightsquigarrow} Y_0 \qquad X \overset{as\,1}{\rightsquigarrow} Y_1}{X \mid !X \overset{b}{\rightsquigarrow} X \mid (!X \mid (Y_0 \mid Y_1))}\,(\mathsf{DRL}_3)\left[\exists c.\ \begin{array}{l}\mathsf{sync}\,(as\,0)\,(as\,1)\,c\\ \wedge\ c = b\end{array}\right] \qquad \dfrac{X \overset{as\,0}{\rightsquigarrow} Y_0 \qquad X \overset{as\,1}{\rightsquigarrow} Y_1}{X \mid !X \overset{b}{\rightsquigarrow} Y_0 \mid (!X \mid Y_1)}\,(\mathsf{DRL}_4)\left[\exists c.\ \begin{array}{l}as\,1 = c\ \wedge\\ \mathsf{sync}\,(as\,0)\,c\,b\end{array}\right]$$

$$\dfrac{X \overset{as\,0}{\rightsquigarrow} Y}{!X \overset{b}{\rightsquigarrow} !X \mid Y}\,(\mathsf{DRL}_5)\,[as\ 0 = b] \qquad \dfrac{X \overset{as\,0}{\rightsquigarrow} Y_0 \qquad X \overset{as\,1}{\rightsquigarrow} Y_1}{!X \overset{b}{\rightsquigarrow} !X \mid (Y_0 \mid Y_1)}\,(\mathsf{DRL}_6)\,[\mathsf{sync}\,(as\,0)\,(as\,1)\,b]$$

Remarks. (1) Because the term $!X$ has only depth 1, the matched derived rules $\mathsf{DRL}_5, \mathsf{DRL}_6$ are essentially the primitive rules $\mathsf{REPL}, \mathsf{REPLS}$. Moreover, DRL_1 was obtained by a single (backwards) application of the rule PARL.

(2) Each of $\mathsf{DRL}_2, \mathsf{DRL}_3, \mathsf{DRL}_4$ arises from the composition of two primitive rules. For example, DRL_3 is obtained by applying PARR, and then applying REPLS to the resulted hypothesis:

$$\dfrac{\dfrac{X \overset{as\,0}{\rightsquigarrow} Y_0 \qquad X \overset{as\,1}{\rightsquigarrow} Y_1}{!X \overset{c}{\rightsquigarrow} !X|(Y_0 \mid Y_1)}\,(\mathsf{REPLS})\,[\mathsf{sync}\,(as\,0)\,(as\,1)\,c]}{X \mid !X \overset{b}{\rightsquigarrow} X \mid (!X \mid (Y_0 \mid Y_1))}\,(\mathsf{PARR})\,[c = b]$$

The side-condition of DRL_3 is obtained by composing (essentially as relations) the two side-conditions, of PARR and REPLS, yielding existential quantification over c. Of course, the side-conditions of $\mathsf{DRL}_2, \mathsf{DRL}_3, \mathsf{DRL}_4$ can be readily simplified to the equivalent forms $as\ 0 = b$, $\mathsf{sync}\,(as\,0)\,(as\,1)\,b$ and again $\mathsf{sync}\,(as\,0)\,(as\,1)\,b$, but eliminating the existential quantifiers may not be possible in general – recall that side-conditions are *arbitrary* predicates.

The only property we care about concerning elements *drl* of $\mathsf{mdr}_V\,U$ w.r.t. V is that $\mathsf{theYs}(drl)$ are all distinct from the variables of V. On the other hand, concerning the relationship between $\mathsf{mdr}_V\,U$ and U, we have the crucial facts of *soundness* and *completeness* w.r.t. transition:

- For all $drl \in \mathsf{mdr}_V\,U$, *drl* is *sound*, i.e.: for all τ :: **var** $\Rightarrow$ **term**, *as* :: **nat** $\Rightarrow$ **act**, and b :: **act**, if $\tau((\mathsf{theXXs}\ drl)!j) \overset{as\,j}{\rightsquigarrow} \tau((\mathsf{theYs}\ drl)!j)$ for all $j < \mathsf{length}(\mathsf{theYs}\ drl)$ and $\mathsf{side}\ drl\ as\ b$ holds, then $(\mathsf{theS}\ drl)[\tau] \overset{b}{\rightsquigarrow} (\mathsf{theT}\ drl)[\tau]$.

- $\mathsf{mdr}_V\,U$ is *complete* for inference of transitions with sources that match U, i.e.: for all τ :: **var** $\Rightarrow$ **term**, b :: **act** and Q :: **term** such that $U[\tau] \overset{b}{\rightsquigarrow} Q$, there exist $drl \in \mathsf{mdr}_V\,U$, τ' :: **var** $\Rightarrow$ **term** and *as* :: **nat** $\Rightarrow$ **act** such that:

— τ' coincides with τ on $\mathsf{vars}\ U$ (hence $U[\tau] = U[\tau']$);

— $\tau'((\mathsf{theXXs}\ drl)!j) \overset{as\,j}{\rightsquigarrow} \tau'((\mathsf{theYs}\ drl)!j)$ for all $j < \mathsf{length}(\mathsf{theYs}\ drl)$;

— $\mathsf{side}\ drl\ as\ b$ holds;

— $(\mathsf{theT}\ drl)[\tau'] = Q$ (and also, remember that $\mathsf{theS}\ drl = U$).

Deduction of universal bisimulation. An *equation* will be simply a pair of terms, written $U \cong V$, and we write **equation** for the type of equations. (Note that **rel** is the same as **equation** <u>**set**</u>.) Our goals will consist of pairs (set of equations) – equation, where all equations shall be thought of as being *universally quantified*. We shall mostly use S, T, U, V for terms thought of as *patterns*, and P, Q, R for terms thought of as *instances*.

Given U, U' :: **term**, $G :: \mathsf{mdr}_{U'}\, U \Rightarrow \mathsf{mdr}_U\, U'$, and $g :: \prod_{drl \in \mathsf{mdr}_{U'}\, U} \{0, \ldots, \mathsf{length}(\mathsf{theXXs}(G\ \ drl)) - 1\} \Rightarrow \{0, \ldots, \mathsf{length}(\mathsf{theXXs}\ \ drl) - 1\}$, we define the predicate $\mathsf{simul}\ U\ U'\ G\ g$, read "$U$ is *(one-step-)simulated* by U' via G and g", to mean that, for all $drl \in \mathsf{mdr}_U\, U'$, the following holds: Assume drl has the form

$$\frac{XX_0 \overset{as\ 0}{\rightsquigarrow} Y_0\ ,\ \ldots\ , XX_{n-1} \overset{as\ (n-1)}{\rightsquigarrow} Y_{n-1}}{S \overset{b}{\rightsquigarrow} T}\ [\mathsf{side}\ drl\ as\ b] \qquad (*)$$

and $drl' = G\ drl$ has the form

$$\frac{XX'_0 \overset{as\ 0}{\rightsquigarrow} Y'_0\ ,\ \ldots\ , XX'_{n'-1} \overset{as\ (n'-1)}{\rightsquigarrow} Y'_{n'-1}}{S' \overset{b}{\rightsquigarrow} T'}\ [\mathsf{side}\ drl'\ as\ b] \qquad (**)$$

(and therefore $g\ drl :: \{0, \ldots, n' - 1\} \Rightarrow \{0, \ldots, n - 1\}$) Then:

- (1) $XX_{g\ drl\ j} = XX'_j$ (i.e., syntactically equal, as variables) for all $j < n'$.
- (2) $\forall as :: \mathbf{nat} \Rightarrow \mathbf{act}, b :: \mathbf{act}.\ \mathsf{side}\ drl\ as\ b \longrightarrow \mathsf{side}\ (G\ drl)\ (as \circ (g\ drl))\ b$.

Given the rules drl, of the form $(*)$, and drl', of the form $(**)$, and given $h :: \{0, \ldots, n' - 1\} \Rightarrow \{0, \ldots, n - 1\}$, we define $\mathsf{newGoal}\ drl\ drl'\ h$ to be the equation $T \cong T'[(Y'_j / Y_{h\,j})_{j<n'}]$, where $(Y'_j / Y_{h\,j})_{j<n'}$ is a substitution that maps each variable Y'_j to the variable $Y_{h\,j}$ (more accurately, to the term $\mathsf{Var}\ Y_{h\,j}$).

simul and $\mathsf{newGoal}$ will work in tandem in our deduction system as follows: Given a goal $U \cong U'$, we wish to prove U and U' universally bisimilar. For this, we should show that, for any continuation of an instance of U, there exists a bisimilar continuation of an instance of U' (and vice versa, but next we ignore the "vice versa" part). By the completeness of mdr, any transition of an instance of U is given by a derived rule drl in $\mathsf{mdr}_{U'}\, U$. By the soundness of mdr, for finding a transition of an instance of U' that simulates that of U, it would suffice to find for drl a derived rule in drl' which is possible whenever drl is possible. Thus, we first need a map $G :: \mathsf{mdr}_{U'}\, U \Rightarrow \mathsf{mdr}_U\, U'$ (giving the drl' for each $drl \in \mathsf{mdr}_{U'}\, U$), and then, for each drl, a justification of the possibility of $G\ drl$ in terms of that of drl. Now, possibility of (a transition along) a derived rule is given by its (formal) hypotheses and its side conditions. Hence, a justification of the possibility of $G\,drl$ in terms of the possibility of drl can be given by a map from the hypotheses of $G\,drl$ to those of drl that *preserves the sources* (which are variables) and *yields an implication between the side conditions* – this is formally achieved by a function $g :: \prod_{drl \in \mathsf{mdr}_{U'}\, U} \{0, \ldots, \mathsf{length}(\mathsf{theXXs}(G\ \ drl)) - 1\} \Rightarrow \{0, \ldots, \mathsf{length}(\mathsf{theXXs}\ \ drl) - 1\}$ that, together with G, satisfies the conditions defining $\mathsf{simul}\ U\ U'\ G\ g$. Moreover, we have to prove that, for each combination $(drl, G\ \ drl)$, the resulted continuations of the presumptive instances of U and U' are again bisimilar – we obtain a $\mathsf{newGoal}\ drl\ (G\ drl)\ (g\ drl)$ for each such combination (note that generating this new goal has to take into consideration the dispatching of formal hypotheses performed by $g\ drl$, meaning that we also have to substitute some "Ys"). Finally, the incremental nature of our coinduction (inherited from the previous section) shows up: for proving each of the new goals, we may *assume* the old goal, $U \cong U'$.

We are led to the deduction relation $\vdash$ :: **equation** $\underline{\mathbf{set}}$ $\Rightarrow$ **equation** $\Rightarrow$ **bool** (with infix notation), defined inductively by the following clauses:

$$\frac{\cdot}{\theta \vdash U \cong U'}\,(\mathsf{Eqnl})[\theta \cup \mathsf{bis} \vdash_{\mathrm{eq}} U \cong U']$$

$$\frac{\begin{array}{c}\forall drl \in \mathsf{mdr}_{U'}\, U.\ \theta \cup \{U \cong U'\} \vdash \mathsf{newGoal}\ drl\ (G\ drl)\ (g\ drl) \\ \forall drl' \in \mathsf{mdr}_{U}\, U'.\ \theta \cup \{U \cong U'\} \vdash \mathsf{newGoal}\ drl'\ (G'\ drl')\ (g'\ drl')\end{array}}{\theta \vdash U \cong U'}\,(\mathsf{Coind})\begin{bmatrix}\mathsf{simul}\ U\ U'\ G\ g \\ \mathsf{simul}\ U'\ U\ G'\ g'\end{bmatrix}$$

In the side-condition at (Eqnl), $\vdash_{\mathrm{eq}}$ is standard equational-logic deduction. We include bis among the hypotheses, because we wish to allow any known facts about bisimilarity to "help" $\vdash$-deduction, including facts obtained by means other than $\vdash$. Again, due to circularity (moving goals to the hypotheses), a rule like (Coind) cannot be sound in itself, but again we have global soundness:

Theorem 3. *If* $\emptyset \vdash U \cong U'$*, then* $(U, U') \in \mathsf{ubis}$.

Proof sketch. We use the soundness of $\Vdash$ (Theorem 1) together with the rules defining $\vdash$ being simulated by those defining $\Vdash$. Namely, we show, by induction on $\vdash$, that $\theta \vdash U \cong U'$ implies $\mathsf{sstvsmCl}(\theta) \Vdash \mathsf{sstvsmCl}(\{(U, U')\})$, where $\mathsf{sstvsmCl} :: \mathbf{rel} \Rightarrow \mathbf{rel}$ gives the *substitutive and symmetric closure of a relation*, i.e., $\mathsf{sstvsmCl}(\xi) = \{(S[\sigma], T[\sigma]).\ \sigma :: \mathbf{var} \Rightarrow \mathbf{term},\ (S, T) \in \xi\ \vee\ (T, S) \in \xi\}$.

If $\theta \vdash U \cong U'$ followed by an application of (Eqnl), then $\mathsf{sstvsmCl}(\theta) \Vdash \mathsf{sstvsmCl}(\{(U, U')\})$ follows applying the $\Vdash$-clause (Ax), since the equational closure coincides with the substitutive symmetric closure of the congruence closure.

Assume now $\theta \vdash U \cong U'$ followed by (Coind), meaning that there exist G, g, G', g' such that: **(i)** $\mathsf{simul}\ U\ U'\ G\ g$; **(ii)** $\forall drl \in \mathsf{mdr}_{U'}\, U.\ \theta \cup \{U \cong U'\} \vdash \mathsf{newGoal}\ drl\ (G\ drl)\ (g\ drl)$; **(iii)** $\mathsf{simul}\ U'\ U\ G'\ g'$; **(iv)** $\forall drl' \in \mathsf{mdr}_{U}\, U'.\ \theta \cup \{U \cong U'\} \vdash \mathsf{newGoal}\ drl'\ (G'\ drl')\ (g'\ drl')$. Then, by the induction hypothesis:

- $\forall drl \in \mathsf{mdr}_{U'}\, U.\ \mathsf{sstvsmCl}(\theta \cup \{U \cong U'\}) \Vdash \mathsf{sstvsmCl}(\{\mathsf{newGoal}\ drl\ (G\ drl)\ (g\ drl)\})$.
- $\forall drl' \in \mathsf{mdr}_{U}\, U'.\ \mathsf{sstvsmCl}(\theta \cup \{U \cong U'\}) \Vdash \mathsf{sstvsmCl}(\mathsf{newGoal}\ drl'\ (G'\ drl')\ (g'\ drl'))$.

Let $\theta' = \mathsf{sstvsmCl}(\{(U, U')\})$ and let $\theta'' = \{\mathsf{newGoal}\ drl\ (G\ drl)\ (g\ drl).\ drl \in \mathsf{mdr}_{U'}\, U\} \cup \{\mathsf{newGoal}\ drl'\ (G'\ drl')\ (g'\ drl').\ drl' \in \mathsf{mdr}_{U}\, U'\}$. The crucial thing to notice is that, since $\mathsf{simul}\ U\ U'\ G\ g$ and $\mathsf{simul}\ U'\ U\ G'\ g'$ hold, $\mathsf{sstvsmCl}(\{(U, U')\}) \subseteq \mathsf{Retr}\ \theta'$ also holds – and the paragraph right before introducing $\vdash$ can be regarded as an informal justification for why this is true. Therefore, θ'' is an "interpolant" for applying the $\Vdash$-clause (Coind). Indeed, applying the $\Vdash$-clause (Split) to (1) and (2), we obtain $\theta' \cup \theta \Vdash \theta''$ and then, by the $\Vdash$-clause (Coind), we obtain $\theta \Vdash \theta'$, as desired. □

Running example (finished). We are now ready to make rigorous the proof of $\forall P :: \mathbf{term}.\ (P|!P, !P) \in \mathsf{bis}$ presented in the introduction. Consider the following four proof trees of depth 0 (later referred to as Pf_1, Pf_2, Pf_3, Pf_4) where we list the side-conditions for (Eqnl) as hypotheses:

$$\dfrac{\{X|!X \cong !X\} \cup \mathsf{bis} \vdash_{\mathrm{eq}} Y|!X \cong !X|Y}{\{X|!X \cong !X\} \vdash Y|!X \cong !X|Y}\,(\mathsf{Eqnl})$$
$$\dfrac{\{X|!X \cong !X\} \cup \mathsf{bis} \vdash_{\mathrm{eq}} X|(!X|Y) \cong !X|Y}{\{X|!X \cong !X\} \vdash X|(!X|Y) \cong !X|Y}\,(\mathsf{Eqnl})$$
$$\dfrac{\{X|!X \cong !X\} \cup \mathsf{bis} \vdash_{\mathrm{eq}} X|(!X|(Y_0|Y_1)) \cong !X|(Y_0|Y_1)}{\{X|!X \cong !X\} \vdash X|(!X|(Y_0|Y_1)) \cong !X|(Y_0|Y_1)}\,(\mathsf{Eqnl})$$
$$\dfrac{\{X|!X \cong !X\} \cup \mathsf{bis} \vdash_{\mathrm{eq}} Y_0|(!X|Y_1) \cong !X|(Y_0|Y_1)}{\{X|!X \cong !X\} \vdash Y_0|(!X|Y_1) \cong !X|(Y_0|Y_1)}\,(\mathsf{Eqnl})$$

Then our final proof (tree) is:

$$\dfrac{Pf_1 \quad Pf_2 \quad Pf_3 \quad Pf_4}{\emptyset \vdash X|!X \cong !X}\,(\mathsf{Coind})$$

Explanations. At (Coind), we took:
- G to map DRL_1 and DRL_2 to DRL_5, and to map DRL_3 and DRL_4 to DRL_6;
- g DRL_1 and g DRL_2 to be the identity on $\{0\}$, and g DRL_3 and g DRL_4 to be the identity on $\{0, 1\}$;
- G' to map DRL_5 to DRL_1, and to map DRL_6 to DRL_3;
- g' DRL_5 to be the identity on $\{0\}$, and g' DRL_6 to be the identity on $\{0, 1\}$.
(Note that any function G' mapping DRL_5 to either DRL_1 or DRL_2 and DRL_6 to either DRL_3 or DRL_4 together with g' as above would lead to a valid proof.)

Here is why we end up with the above four proof tasks after applying (Coind):

$\mathsf{newGoal}\,\mathsf{DRL}_1(G\,\mathsf{DRL}_1)(g\,\mathsf{DRL}_1) = \mathsf{newGoal}\,\mathsf{DRL}_1\,\mathsf{DRL}_5(\lambda i.\,i) = Y|!X \cong !X|Y$;
$\mathsf{newGoal}\,\mathsf{DRL}_2(G\,\mathsf{DRL}_2)(g\,\mathsf{DRL}_2) = \mathsf{newGoal}\,\mathsf{DRL}_2\,\mathsf{DRL}_5(\lambda i.\,i) = X|(!X|Y) \cong !X|Y$;
$\mathsf{newGoal}\,\mathsf{DRL}_3(G\,\mathsf{DRL}_3)(g\,\mathsf{DRL}_3) = \mathsf{newGoal}\,\mathsf{DRL}_3\mathsf{DRL}_6(\lambda i.\,i) = X|(!X|(Y_0|Y_1)) \cong !X|(Y_0|Y_1)$;
$\mathsf{newGoal}\,\mathsf{DRL}_4(G\,\mathsf{DRL}_4)(g\,\mathsf{DRL}_4) = \mathsf{newGoal}\,\mathsf{DRL}_4\,\mathsf{DRL}_6(\lambda i.\,i) = Y_0|(!X|Y_1) \cong !X|(Y_0|Y_1)$;
$\mathsf{newGoal}\,\mathsf{DRL}_5(G'\mathsf{DRL}_5)(g'\mathsf{DRL}_5) = \mathsf{newGoal}\,\mathsf{DRL}_5\,\mathsf{DRL}_1(\lambda i.\,i) = Y|!X \cong !X|Y$;
$\mathsf{newGoal}\,\mathsf{DRL}_6(G'\mathsf{DRL}_6)(g'\mathsf{DRL}_6) = \mathsf{newGoal}\,\mathsf{DRL}_6\mathsf{DRL}_3(\lambda i.\,i) = X|(!X|(Y_0|Y_1)) \cong !X|(Y_0|Y_1)$.

The side-conditions of (Coind) are immediately checkable. E.g., for $\mathsf{simul}\ (X|!X)\ (!X)\ G\ g$, we need to check the following trivial facts:
- w.r.t. condition (1) (in the definition of simul): that $X = X$.
- w.r.t. condition (2): that each of the following are pairwise equivalent:
— $as\ 0 = b$ and $as\ 0 = b$;
— $\exists c.\ as\ 0 = c \wedge c = b$ and $as\ 0 = b$;
— $\exists c.\ \mathsf{sync}\,(as\ 0)\,(as\ 1)\,c \wedge c = b$ and $\mathsf{sync}\,(as\ 0)\,(as\ 1)\,b$;
— $\exists c.\ as\ 1 = c \wedge \mathsf{sync}\,(as\ 0)\,c\,b$ and $\mathsf{sync}\,(as\ 0)\,(as\ 1)\,b$.

At (Eqnl) in all the four immediate subtrees of the main proof tree, we considered the fact (assumed previously proved) that $\{X|Y \cong Y|X,\ (X|Y)|Z \cong X|(Y|Z)\} \subseteq \mathsf{bis}$, hence what we really used was equational-logic deduction from $\{X|!X \cong !X,\ X|Y \cong Y|X,\ (X|Y)|Z \cong X|(Y|Z)\}$, which easily discharges the equational side-conditions of the axioms, finalizing the proof.

The above proof does not display any non-trivial "dispatch" function g in the (Coind) rule application. In general however, it is not guaranteed that the formal hypotheses of two obtained derived rules (from the two terms of the goal) that one wishes to pair come in the same order, nor that these rules have the same number of hypotheses. (See the proof of commutativity of "|" from App. B in [28].)

5 Concluding Remarks

We have developed and formalized in Isabelle a proof system for process algebra where bisimilarity is proved *incrementally*, while exploring and expanding the goal, without requiring an a priori constructed bisimulation relation. Our results apply to a wide class of process algebras.

Related work. Unique fixpoint induction for CCS and its variants [23,17,25] is an early notion of proof-theoretic circularity for coinduction applicable to situations where circularity is *explicit* in the SOS by means of (guarded) fixpoint equations. We conjecture that unique fixpoint induction in an instance of our incremental coinduction.

We had two major sources of inspiration. First, the idea of *circular coinduction* (CC for short) in the context of algebraic specifications. It was introduced in [14] in the behavioral specification language BOBJ [1], and then also implemented axiomatically in (generic) Isabelle under the "supervision" of the CoCASL specification language [16] and in Maude [9] as the circular coiductive prover CIRC [20,19,31]. A comparison of our proof system with CC is somewhat difficult to sketch, as it has to deal with different technical settings and to balance the advantages of both generality and specialization. To simplify the discussion, we shall implicitly assume a back-and-forth translation between SOS specifications and the coalgebraic and behavioral specifications required by the CC settings. Our proof system is in a sense more general and in a sense more specialized.

It is more general in that it applies to *nondeterministic processes*, not handled by CC (e.g., the running example in this paper is not approachable in CC, not even interactively). On the other hand, CIRC, based on rewriting logic, could employ the results presented here in order to extend CC with nondeterminism.[2] Also, CoCASL as a specification language has the expressive power required to deal with process algebra and nondeterminism, hence to support a version of CC for nondeterministic systems. However, it is the determinism of CC that allows for partial automation, admirably illustrated by CIRC. (Our formalized system, once fine-tuned into a tool, will also allow automation for deterministic, and, to some extent, finitely-branching cases – see below the discussion on future work.)

It is more specialized in that the *deterministic* instances of our setting are more restricted than what CC can handle (in particular, e.g., deterministic lookahead, not approachable here, is unproblematic in CC). On the other hand, our powerful coinduction "up to", underneath arbitrary contexts (not supported by CC) is possible precisely because of this restriction.

[2] Which is not to say our proof system is a minor variation of CC – nondeterminism (technically, the interplay between our rules (Split) and (Coind) from Sec. 3) represented the main difficulty in our soundness proof.

Finally, our coinductive technique is presented in a logical form, as a *proof system*, like in [31], and not as an *algorithm* like in the other cited works on CC. In [31], logical form is achieved through the introduction of so-called *freezing operators*, hard to justify logically – with this respect, our proof system has the advantage of "purity".[3] (Here we should also remark some less related work: circular systems in logical form were also developed in [7,10] for first-order logic and the μ-calculus, respectively.)

The second major source of inspiration was the notion of coinduction proofs up to bisimilarity and arbitrary contexts, introduced in [11,24] and developed in [34,35]. This idea also appears in a general coalgebraic setting in [5] and is illustrated by extensive examples in, e.g., [33]. The convenience of performing unrestricted equational reasoning relies essentially on the "up to" coinduction principle, Theorem 1.

Other related work includes frameworks for *bisimilarity of open terms* in [29,8,4] (also building on the seminal work from [11]), where open terms are considered universally quantified, as we do in this paper for universal bisimilarity. Our soundness result for $\vdash$ w.r.t. universal bisimilarity, Theorem 3, could have been more sharply phrased: on one hand, as a soundness result w.r.t. the notion of *bisimulation under formal hypotheses* from [11,29]; on the other, w.r.t. to the relation from [4] (which is essentially universal bisimilarity in any conservative extension of the SOS system). All works cited in this paragraph discuss non-incremental proof systems, where the desired bisimulation relation needs to be fed by the user.

Descriptions of more or less automatic software tools for proving bisimilarity in process algebra abound in the literature – see [18,21] for overviews. While most of these tools are dedicated to (and optimized for) particular process algebras (and many to *finite-state* systems), ECRINS [12] is based precisely on generic process algebra in de Simone format, meaning that the results of this paper on incremental coinduction apply directly to that setting (and, interestingly, a form of coinduction that "attempts to add more couples to the [previously specified] relation" is indicated in [12] as a direction for future research, to our knowledge not pursued so far). Finally, in Coq [2], the interaction between its general-purpose support for building proofs and its coinductive types (as illustrated, e.g., in [13]) also leads to a form of incremental coinduction whose relationship with our approach is yet to be understood.

Future work. The de Simone SOS format is already fairly general, covering a wide range of process algebras, including CCS-like process algebras with *guarded recursion*, and even de Simone systems under weak bisimilarity (since for them weak bisimilarity can be regarded as strong bisimilarity for trace-based de Simone systems, as suggested in [27]). An extension of our incremental coinductive technique to more general formats such as GSOS [6] or tyft/tyxt [15] is of course

[3] In a sense, what these freezing operators do is to guard *against* coinduction up-to, not sound in general. So again, our logical system achieves convenience because of specialization, i.e., by sacrificing some generality.

desirable. Another direction for generalization is the allowance of bindings in the syntax of terms, including π-calculus-like bindings featuring scope extrusion (thus generalizing HOL-based settings for π-calculus such as [22,30]).

In our proof system for universal bisimilarity, $\vdash$, one has to come up with suitable dispatch functions G, g, G', g' at each application of the coinduction rule (Coind), and therefore (Coind) is not syntax-directed per se, hence not trivially automatable – this is inherent in the hardness of bisimilarity in our general setting. However, (Coind) does allow to decompose the goal symbolically, without asking the user to decide for a global bisimilarity candidate – instead, assisted by the powerful Isabelle classical reasoner and simplifier (able to discharge the typically simple goals resulted from chaining side-conditions), the user can explore various choices of the dispatch functions by analyzing the derived rules. Moreover, for systems with finite number of rules one can write an Isabelle tactic that tries all the combinations of dispatch functions. While this would require exponential time, it may still be feasible for cases of interest, since the time-complexity is a function of the *symbolic* branching of process patterns (determined by logical formulas obtained from side-conditions), and not of the actual branching of processes given by all possible instances of the rules. For the general case, we should aim at organizing our formalization (by means of proof tactics and pretty-printers) into a partly-interactive partly-automatic tool. The advantages of such a tool will of course include the generality of its scope and the fact that, unlike most other tools, it would be (a priori) *formally certified.*

Acknowledgments. We thank the reviewers for the careful and insightful reading of this paper and for pointing out related work. The first author is grateful to Grigore Roşu for many hours of discussion on his circular coinduction technique.

References

1. BOBJ, `http://cseweb.ucsd.edu/groups/tatami/bobj`
2. The Coq proof assistant, `http://coq.inria.fr`
3. Isabelle, `http://www.cl.cam.ac.uk/research/hvg/Isabelle`
4. Aceto, L., Cimini, M., Ingolfsdottir, A.: A bisimulation-based method for proving the validity of equations in GSOS languages. To appear in Electr. Proc. Theor. Comput. Sci.
5. Bartels, F.: Generalised coinduction. Math. Struct. Comp. Sci. 13(2), 321–348 (2003)
6. Bloom, B., Istrail, S., Meyer, A.R.: Bisimulation can't be traced. J. ACM 42(1), 232–268 (1995)
7. Brotherston, J.: Cyclic proofs for first-order logic with inductive definitions. In: Beckert, B. (ed.) TABLEAUX 2005. LNCS (LNAI), vol. 3702, pp. 78–92. Springer, Heidelberg (2005)
8. Bruni, R., de Frutos-Escrig, D., Martí-Oliet, N., Montanari, U.: Bisimilarity congruences for open terms and term graphs via Tile Logic. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 259–274. Springer, Heidelberg (2000)

9. Clavel, M., Durán, F.J., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Quesada, J.F.: The Maude system. In: Narendran, P., Rusinowitch, M. (eds.) RTA 1999. LNCS, vol. 1631, pp. 240–243. Springer, Heidelberg (1999)
10. Dam, M., Gurov, D.: μ-calculus with explicit points and approximations. J. Log. Comput. 12(2), 255–269 (2002)
11. de Simone, R.: Higher-level synchronizing devices in MEIJE-SCCS. Theor. Comput. Sci. 37, 245–267 (1985)
12. Doumenc, G., Madelaine, E., de Simone, R.: Proving process calculi translations in ECRINS: The pureLOTOS → MEIJE example. Technical Report RR1192, INRIA (1990), http://hal.archives-ouvertes.fr/inria-00075367/en/
13. Giménez, E.: An application of co-inductive types in Coq: Verification of the alternating bit protocol. In: Berardi, S., Coppo, M. (eds.) TYPES 1995. LNCS, vol. 1158, pp. 135–152. Springer, Heidelberg (1996)
14. Goguen, J.A., Lin, K., Roşu, G.: Circular coinductive rewriting. In: ASE 2000, pp. 123–132 (2000)
15. Groote, J.F., Vaandrager, F.: Structured operational semantics and bisimulation as a congruence. Inf. Comput. 100(2), 202–260 (1992)
16. Hausmann, D., Mossakowski, T., Schröder, L.: Iterative circular coinduction for coCASL in Isabelle/HOL. In: Cerioli, M. (ed.) FASE 2005. LNCS, vol. 3442, pp. 341–356. Springer, Heidelberg (2005)
17. Hennessy, M., Lin, H.: Proof systems for message-passing process algebras. Formal Asp. Comput. 8(4), 379–407 (1996)
18. Inverardi, P., Priami, C.: Automatic verification of distributed systems: The process algebra approach. Formal Methods in System Design 8(1), 7–38 (1996)
19. Lucanu, D., Goriac, E.-I., Caltais, G., Roşu, G.: CIRC: A behavioral verification tool based on circular coinduction. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) CALCO 2009. LNCS, vol. 5728, pp. 433–442. Springer, Heidelberg (2009)
20. Lucanu, D., Roşu, G.: CIRC: A circular coinductive prover. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) CALCO 2007. LNCS, vol. 4624, pp. 372–378. Springer, Heidelberg (2007)
21. Madelaine, E.: Verification tools from the CONCUR project, http://www-sop.inria.fr/meije/papers/concur-tools
22. Melham, T.F.: A mechanized theory of the pi-calculus in HOL. Nord. J. Comput. 1(1), 50–76 (1994)
23. Milner, R.: A complete inference system for a class of regular behaviours. J. Comput. Syst. Sci. 28(3), 439–466 (1984)
24. Milner, R.: Communication and concurrency. Prentice-Hall, Englewood Cliffs (1998)
25. Monroy, R., Bundy, A., Green, I.: On process equivalence = equation solving in ccs. J. Autom. Reasoning 43(1), 53–80 (2009)
26. Mousavi, M.R., Reniers, M.A., Groote, J.F.: SOS formats and meta-theory: 20 years after. Theor. Comput. Sci. 373(3), 238–272 (2007)
27. Popescu, A.: Weak bisimilarity coalgebraically. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) CALCO 2009. LNCS, vol. 5728, pp. 157–172. Springer, Heidelberg (2009)
28. Popescu, A., Gunter, E.L.: Incremental pattern-based coinduction for process algebra and its Isabelle formalization. Technical Report, University of Illinois, http://hdl.handle.net/2142/14858
29. Rensink, A.: Bisimilarity of open terms. Inf. Comput. 156(1-2), 345–385 (2000)

30. Röckl, C., Hirschkoff, D.: A fully adequate shallow embedding of the π-calculus in Isabelle/HOL with mechanized syntax analysis. J. Funct. Program. 13(2) (2003)
31. Roşu, G., Lucanu, D.: Circular coinduction: A proof theoretical foundation. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) CALCO 2009. LNCS, vol. 5728, pp. 127–144. Springer, Heidelberg (2009)
32. Rutten, J.J.M.M.: Processes as terms: Non-well-founded models for bisimulation. Math. Struct. Comp. Sci. 2(3), 257–275 (1992)
33. Rutten, J.J.M.M.: Elements of stream calculus (an extensive exercise in coinduction). Electr. Notes Theor. Comput. Sci., 45 (2001)
34. Sangiorgi, D.: On the bisimulation proof method. Math. Struct. Comp. Sci. 8(5), 447–479 (1998)
35. Sangiorgi, D., Walker, D.: The π-calculus. A theory of mobile processes, Cambridge (2001)