Unique solutions of contractions, CCS, and their HOL formalisation

Chun Tian
Fondazione Bruno Kessler
Trento, Italy
ctian@fbk.eu

Davide Sangiorgi
Università di Bologna and INRIA
Bologna, Italy
davide.sangiorgi@unibo.it

The unique solution of contractions is a proof technique for bisimilarity that overcomes certain syntactic constraints of Milner's "unique solution of equations" technique. The paper presents an overview of the a rather comprehensive formalisation of the core of the theory of CCS in the HOL theorem prover (HOL4), with a focus towards the theory of unique solutions of contractions. (The formalisation consists of about 20,000 lines of proof scripts in Standard ML.) Some refinements of the theory itself are obtained. In particular we remove the constraints on the occurrences of the sum operator by moving to rooted contraction, that is, the coarsest precongruence contained in the contraction preorder.

1 Introduction

A prominent proof method for bisimulation, put forward by Milner and widely used in his landmark CCS book [10] is the *unique solution of equations*, whereby two tuples of processes are componentwise bisimilar if they are solutions of the same system of equations. This method is important in verification techniques and tools based on algebraic reasoning [3, 14].

In the *weak* case (when behavioural equivalences abstract from internal moves, which practically is the most relevant case), however, Milner's proof method has severe syntactical limitations. A way for overcoming such limitations is to replace equations with special inequations called *contractions* [15]. Contraction is a preorder that, roughly, places some efficiency constraints on processes. Uniqueness of the solution of a system of contractions is defined as with systems of equations: any two solutions must be bisimilar. The difference with equations is in the meaning of a solution: in the case of contractions the solution is evaluated with respect to the contraction preorder, rather than bisimilarity. With contractions, most syntactic limitations of the unique-solution theorem can be removed. One constraint that still remains in [15] is on occurrences of the sum operator, due to the failure of the substitutivity of contraction for such an operator.

The main goal of the work described in this paper is a rather comprehensive formalisation of the core of the theory of CCS in the HOL theorem prover (HOL4), with a focus on the theory of unique solutions of contractions. The formalisation however is not confined to the theory of unique solutions, but embraces most of the core of the theory of CCS [10] (partly because the theory of unique solutions relies on a number of more fundamental results): indeed the formalisation encompasses the basic properties of strong and weak bisimilarity (e.g. the fixed-point and substitutivity properties), their basic algebraic theory, various versions of "bisimulation up to" techniques (e.g., bisimulation up-to bisimilarity), the main properties of the rooted bisimilarity (the congruence induced by weak bisimilarity, also called observation congruence), and of the expansion preorder (an efficiency-like refinement of bisimilarity). Concerning rooted bisimilarity,

Submitted to:
EXPRESS/SOS 2018

the formalisation includes Hennessy and Deng lemmas, and two proofs that rooted bisimilarity is the largest congruence included in bisimilarity: one as in Milner's book, requiring the hypothesis that no processes can use all labels; the other without such hypothesis, essentially formalising van Glabbeek's paper [9] (such a proof however follows the structure of the ordinal numbers, which cannot be handled in HOL, and therefore it is restricted to finite-state processes). Similar theorems are proved for the rooted contraction preorder. In this respect, the work is an extensive experiment with the use of the HOL theorem prover and its most recent developments, including a package for expressing coinductive definitions. The work consists of about 20,000 lines of proof scripts in Standard ML.

From the CCS theory viewpoint, the formalisation has offered us the possibility of further refining the theory of unique solutions. In particular, the existing theory placed limitations on the body of the contractions due to the substitutivity problems of weak bisimilarity and other behavioural relations with respect to the sum operator. We have thus refined the proof technique based on contractions by moving to *rooted contraction*, that is, the coarsest (pre)congruence contained in the contraction preorder. The resulting unique-solution theorem is valid for *rooted bisimilarity* (hence also for bisimilarity itself), and places no constraints on the occurrences of sums.

Another advantage of the formalisation is that we can take advantage of results about different equivalences or preorders that share a similar proof structure. Examples are: the results that rooted bisimilarity and rooted contraction are, respectively, the coarsest congruence contained in weak bisimilarity and the coarsest precongruence contained in the contraction preorder; the result about unique solution of equations for weak bisimilarity that uses the contraction preorder as an auxiliary relation, and other unique solution results (e.g., the one for rooted in which the auxiliary relation is rooted contraction); various forms of enhancement of the bisimulation proof method (the 'up-to' techniques). In these cases, moving between proofs there are only a few places in which the HOL scripts have to be modified. Then the successful termination of the scripts gives us a guarantee that the proof is completed, removing the risk of overlooking or missing details as in hand-written proofs.

Structure of the paper Section 2 presents basic background materials on CCS, including its syntax, operational semantics, bisimilarity and rooted bisimilarity. Section 3 discussed equations and contractions. Section 4 presents rooted contraction and the related unique-solution result for rooted bisimilarity. Section 5 highlight our formalisation in HOL4. Finally, Section 6 and 7 discuss related work, conclusions, and a few directions for future work.

2 CCS

We assume a possibly infinite set of *names* $a, b, \ldots$, which does not contain the special symbol τ, and a set of variables $A, B, \ldots$ for writing recursive behaviours. The class of the CCS processes is inductively built from $\mathbf{0}$ by the operators of prefixing, parallel composition, sum, restriction, and recursion:

$$\begin{array}{rcl} \mu & := & a \;\Big|\; \overline{a} \;\Big|\; \tau \\ P & := & \mathbf{0} \;\Big|\; \mu.P \;\Big|\; P_1 \mid P_2 \;\Big|\; P_1 + P_2 \;\Big|\; (\boldsymbol{\nu} a)P \;\Big|\; \mathtt{rec}\, A.P \;\Big|\; A \end{array}$$

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 and the rule for sum has been omitted).

$$\frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'} \qquad \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{}{\mu.\, P \xrightarrow{\mu} P} \qquad \frac{P \xrightarrow{\mu} P'}{(\boldsymbol{\nu} a)\, P \xrightarrow{\mu} (\boldsymbol{\nu} a)\, P'}\ \mu \neq a, \overline{a} \qquad \frac{P\{\mathtt{rec}\, A.\, P/A\} \xrightarrow{\mu} P'}{\mathtt{rec}\, A.\, P \xrightarrow{\mu} P'}$$

Figure 1: The LTS of CCS

A CCS expression *uses weakly-guarded sums* if all occurrences of the sum operator are of the form $\mu_1.\, P_1 + \mu_2.\, P_2 + \ldots + \mu_n.\, P_n$, for some $n \geq 2$. The *immediate derivatives* of a process P are the elements of the set $\{P' \mid P \xrightarrow{\mu} P' \text{ for some } \mu\ \}$. Some standard notations for transitions: $\Longrightarrow$ is the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\overset{\mu}{\Longrightarrow}$ 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 $P \overset{\widehat{\mu}}{\Longrightarrow} P'$ holds if $P \overset{\mu}{\Longrightarrow} P'$ 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 $\overset{\text{def}}{=}$ for abbreviations. For instance, $P \overset{\text{def}}{=} G$, where G is some expression, means that P stands for the expression G. If $\leq$ is a preorder, then $\geq$ is its inverse (and conversely).

2.1 Bisimilarity and rooted bisimilarity

The equivalences we consider are mainly *weak*, in that they abstract from the number of internal steps performed

Definition 2.1 (bisimilarity). *A process relation $\mathcal{R}$ is a (weak)* bisimulation *if, whenever $P\,\mathcal{R}\,Q$, we have:*

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

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

We sometimes call bisimilarity '*weak*', 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 $Q \xrightarrow{\mu} Q'$. Bisimilarity is not preserved by the sum operator. For this, Milner introduces *observational congruence*, also called *rooted bisimilarity*.

Definition 2.2 (rooted bisimilarity). *Two processes P and Q are* rooted bisimilar*, written $P \approx^{\mathrm{c}} Q$, if we have:*

1. *$P \xrightarrow{\mu} P'$ implies that there is Q' such that $Q \overset{\mu}{\Rightarrow} Q'$ and $P' \approx Q'$;*
2. *the converse of (1) on the actions from Q.*

Theorem 2.3. *$\approx^{\mathrm{c}}$ is a congruence in CCS. Indeed, it is the largest congruence contained in $\approx$.*

3 Equations and contractions

3.1 Systems of equations

Uniqueness of solutions of equations [10] 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 the CCS grammar extended with equation variables.

Definition 3.1. *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.)

Definition 3.2. *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 $P_i \approx E_i[\widetilde{P}]$;*
- *it has* a unique solution for $\approx$ *if whenever $\widetilde{P}$ and $\widetilde{Q}$ are both solutions for $\approx$, then $\widetilde{P} \approx \widetilde{Q}$.*

For instance, the solution of the equation $X = a.\,X$ is the process $R \overset{\mathrm{def}}{=} \mathtt{rec}\, A.\, a.\, A$, and for any other solution P we have $P \approx R$. In contrast, the equation $X = a \mid X$ has solutions that may be quite different, namely any process capable of continuously performing a actions (but that could behave arbitrarily on other actions).

Definition 3.3. *A system of equations $\{X_i = E_i\}_{i \in I}$ is*

- (strongly) 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.*

Theorem 3.4 (unique solution of equations, [10])**.** *A system of guarded and sequential equations has a unique solution for $\approx$.*

To see the need of the sequentiality condition, consider the equation (from [10]) $X = \boldsymbol{\nu} a\,(a.\,X \mid \overline{a})$ where X is guarded but not sequential. Any processes that does not use a is a solution.

3.2 Contractions

The constraints on the unique-solution Theorem 3.4 can be weakened if we move from equations to inequations called *contractions*.

Intuitively, the bisimilarity contraction $\succeq_{\mathrm{bis}}$ is a preorder in which $P \succeq_{\mathrm{bis}} Q$ holds if $P \approx Q$ and, in addition, Q has the *possibility* of being at least as efficient as P (as far as τ-actions performed). 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).

Definition 3.5 (bisimulation contraction)**.** *A process relation* $\mathcal{R}$ *is a* bisimulation contraction *if, whenever* $P \mathcal{R} Q$*,*

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

Bisimilarity contraction*, written* $\succeq_{\text{bis}}$*, 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 3.5, 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* $\succeq_{\text{e}}$ [15] This is a preorder widely used in proof techniques for bisimilarity and that intuitively refines bisimilarity by formalising the idea of 'efficiency' between processes. Clause (1) is the same in the two preorders. But in clause (2) expansion uses $P \overset{\mu}{\Rightarrow} P'$, rather than $P \overset{\widehat{\mu}}{\Rightarrow} P'$; moreover with contraction the final derivatives are simply required to be bisimilar. An expansion $P \succeq_{\text{e}} Q$ tells us that Q is always at least as efficient as P, whereas the contraction $P \succeq_{\text{bis}} Q$ just says that Q has the possibility of being at least as efficient as P.

Example 3.6. *We have* $a \not\succeq_{\text{bis}} \tau.a$*. However,* $a + \tau.a \succeq_{\text{bis}} a$*, as well as its converse,* $a \succeq_{\text{bis}} a + \tau.a$*. Indeed, if* $P \approx Q$ *then* $P \succeq_{\text{bis}} P + Q$*. The last two relations do not hold with* $\succeq_{\text{e}}$*, which explains the strictness of the inclusion* $\succeq_{\text{e}} \subseteq \succeq_{\text{bis}}$*.*

As bisimilarity and expansion, contraction is preserved by all operators but sum.

3.3 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 3.7. *Given a system of contractions* $\{X_i \succeq E_i\}_{i \in I}$*, we say that:*

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

Definition 3.8. *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.*

The system use weakly-guarded sums *if each* E_i *only makes use of guarded sums.*

Lemma 3.9. *Suppose* $\widetilde{P}$ *and* $\widetilde{Q}$ *are solutions for* $\succeq_{\text{bis}}$ *of a system of weakly-guarded contractions that uses weakly-guarded sums. For any context* C *that uses weakly-guarded sums, if* $C[\widetilde{P}] \overset{\mu}{\Rightarrow} R$*, then there is a context* C' *that uses weakly-guarded sums such that* $R \succeq_{\text{bis}} C'[\widetilde{P}]$ *and* $C[\widetilde{Q}] \overset{\widehat{\mu}}{\Rightarrow} \approx C'[\widetilde{Q}]$*.*

Proof. [Sketch] Let n be the length of the transition $C[\widetilde{P}] \stackrel{\mu}{\Rightarrow} R$ (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; moreover all the contexts use weakly-guarded sums.

We have $C[\widetilde{P}] \succeq_{\mathrm{bis}} C''[\widetilde{P}]$, and $C[\widetilde{Q}] \succeq_{\mathrm{bis}} C''[\widetilde{Q}]$, by the substitutivity properties of $\succeq_{\mathrm{bis}}$ (we exploit here the syntactic constraints on sums). Moreover, since each hole of the context C'' is underneath at least n prefixes, applying the definition of $\succeq_{\mathrm{bis}}$ on the transition $C[\widetilde{P}] \stackrel{\mu}{\Rightarrow} R$, we infer the existence of C' such that $C''[\widetilde{P}] \stackrel{\widehat{\mu}}{\Rightarrow} C'[\widetilde{P}] \preceq_{\mathrm{bis}} R$ and $C''[\widetilde{Q}] \stackrel{\widehat{\mu}}{\Rightarrow} C'[\widetilde{Q}]$. Finally, again applying the definition of $\succeq_{\mathrm{bis}}$ on $C[\widetilde{Q}] \succeq_{\mathrm{bis}} C''[\widetilde{Q}]$, we derive $C[\widetilde{Q}] \stackrel{\widehat{\mu}}{\Rightarrow}\approx C'[\widetilde{Q}]$. □

Theorem 3.10 (unique solution of contractions for $\approx$)**.** *A system of weakly-guarded contractions that uses weakly-guarded sums 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 that uses weakly-guarded sums, then the relation

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

is a bisimulation, exploiting Lemma 3.9. □

4 Rooted contraction

The unique solution theorem of Section 3.3 requires a constrained syntax for sums, due to the congruence and precongruence problems of bisimilarity and contraction with such operator. We show here that the constraints can be removed by moving to the induced congruence and precongruence, the latter called *rooted contraction*.

Definition 4.1 (rooted contraction)**.** *Two processes P and Q are in the* rooted contraction*, written $P \succeq^{\mathrm{c}}_{\mathrm{bis}} Q$, if*

1. *$P \xrightarrow{\mu} P'$ implies that there is Q' with $Q \xrightarrow{\mu} Q'$ and $P' \succeq_{\mathrm{bis}} Q'$;*
2. *$Q \xrightarrow{\mu} Q'$ implies that there is P' with $P \stackrel{\mu}{\Rightarrow} P'$ and $P' \approx Q'$.*

The definition of rooted contraction adapts the definition of rooted bisimilarity on top of that of the contraction preorder $\succeq_{\mathrm{bis}}$.

Theorem 4.2. *Relation $\succeq^{\mathrm{c}}_{\mathrm{bis}}$ is a precongruence in CCS. Indeed it is the largest precongruence contained in $\succeq_{\mathrm{bis}}$.*

The proof of this result is along the lines of the analogous result for rooted bisimilarity with respect to bisimilarity.

For a system of contraction, the meaning of 'solution for $\succeq^{\mathrm{c}}_{\mathrm{bis}}$' and of *a unique solution for* $\approx^{\mathrm{c}}$ is the expected one — just replace in Definition 3.7 the preorder $\succeq_{\mathrm{bis}}$ with $\succeq^{\mathrm{c}}_{\mathrm{bis}}$, and the equivalence $\approx$ with $\approx^{\mathrm{c}}$.

For the rooted relations, the analogous of Lemma 3.9 and of Theorem 3.10 can now be stated without constraints on the sum operator. The schema of the proofs is also the same; the substitutivity properties of $\succeq^{\mathrm{c}}_{\mathrm{bis}}$ and $\approx^{\mathrm{c}}$ allow us to avoid issues with the sum operator. Some care is needed in Theorem 4.4 to make sure that the final result is about $\approx^{\mathrm{c}}$, rather than (the weaker) $\approx$.

Lemma 4.3. *Suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\succeq^{\mathrm{c}}_{\mathrm{bis}}$ of a system of weakly-guarded contractions. For any context C, if $C[\widetilde{P}] \overset{\mu}{\Rightarrow} R$, then there is a context C' such that $R \succeq_{\mathrm{bis}} C'[\widetilde{P}]$ and $C[\widetilde{Q}] \overset{\mu}{\Rightarrow}\approx C'[\widetilde{Q}]$.*

Theorem 4.4 (Unique solution of contractions for $\approx^{\mathrm{c}}$)**.** *A system of weakly-guarded contractions has a unique solution for $\approx^{\mathrm{c}}$.*

5 Formalisation

We highlight here a formalisation of CCS in the HOL theorem prover (HOL4) [16], including the new concepts and theorems proposed in the first half of this paper. The whole formalisation can be found in [17] or in in the official example folder of HOL4 source code[1]. The work consists of about 20,000 lines of proof scripts in Standard ML; it develops some pioneering work by Nesi [11], in 1992-1995.

Higher Order Logic (or "HOL Logic") [1] stands for simple-typed λ-calculus plus Hilbert choice operator, axiom of infinity, and rank-1 polymorphism (type variables). HOL4 implemented the original HOL Logic, while some other theorem provers in HOL family (e.g. Isabelle/HOL) have certain extensions. Indeed the HOL Logic has considerable simpler logic foundations than most other theorem provers. As a result, formal theories in HOL can be easily migrated (sometimes even automatically) into other theorem provers.

HOL4 is written in Standard ML (ML for abbreviation), which plays three roles here:

1. It serves as the underlying implementation language for the core HOL engine;
2. it is used to implement tactics (and tacticals) for writing proofs;
3. it is used as the command language of the HOL interactive shell.

Further, in the same language HOL4 users can write complex automatic verification tools by calling HOL's interactive theorem proving facilities.

In this formalisation we consider only single equations/contractions. This simplifies the required proof steps in HOL while enhancing the reading of the scripts.

5.1 CCS and its transitions by SOS rules

In our formalisation of CCS, the type "β `Label`" (β is a type variable) accounts for visible actions, divided into input and output actions:

```
val _ = Datatype ‘Label = name ’b | coname ’b‘;
```

The type "β `Action`" is the union of all visible and invisible actions. The cardinality of "β `Action`" (and therefore of all CCS types built on top of it) depends on the choice of the initial type β.

The core datatype "`(`α`,` β`) CCS`", accounting for the CCS syntax, is then defined as an inductive datatype in HOL, based on its Datatype package (here `'a` is the type variable α for recursion variables, `'b` is the type variable β for actions):

```
val _ = Datatype ‘CCS = nil
                      | var ’a
```

[1] `https://github.com/HOL-Theorem-Prover/HOL`

```
    | prefix ('b Action) CCS
    | sum CCS CCS
    | par CCS CCS
    | restr (('b Label) set) CCS
    | relab CCS ('b Relabeling)
    | rec 'a CCS`;
```

We have added some grammar support, using HOL powerful 'pretty printer', to represent CCS processes in more readable forms (see the 'HOL (abbrev.)' column of Table 1; the table summarizes the main syntactic notations for CCS). For the restriction operator, we have chosen to allow a set of names as a parameter, rather than a single name as in the ordinary CCS syntax; this simplifies the manipulation of processes with different orders of nested restrictions.

Operator	**CCS Notation**	**HOL term**	**HOL (abbrev.)**
nil	$\mathbf{0}$	`nil`	`nil`
prefix	$u.P$	`prefix u P`	$u\texttt{..}P$
sum	$P+Q$	`sum P Q`	P + Q
parallel	$P \mid Q$	`par P Q`	$P \parallel Q$
restriction	$(\nu\, L)\, P$	`restr L P`	$\nu\ L\ P$
recursion	$\texttt{rec}\, A.P$	`rec A P`	`rec` $A\ P$
input action	a	`label (name a)`	`In` a
output action	$\overline{a}$	`label (coname a)`	`Out` a

Table 1: Syntax of CCS operators

The transition semantics of CCS processes follows the Structural Operational Semantics (SOS) rules in Figure 1:

$\vdash u\texttt{..}P\ -u\rightarrow\ P$ [PREFIX]
$\vdash P\ -u\rightarrow\ P' \Rightarrow P + Q\ -u\rightarrow\ P'$ [SUM1]
$\vdash P\ -u\rightarrow\ P' \Rightarrow Q + P\ -u\rightarrow\ P'$ [SUM2]
$\vdash P\ -u\rightarrow\ P' \Rightarrow P \parallel Q\ -u\rightarrow\ P' \parallel Q$ [PAR1]
$\vdash P\ -u\rightarrow\ P' \Rightarrow Q \parallel P\ -u\rightarrow\ Q \parallel P'$ [PAR2]
$\vdash P\ -\texttt{label}\ l\rightarrow\ P' \land Q\ -\texttt{label (COMPL}\ l\texttt{)}\rightarrow\ Q' \Rightarrow P \parallel Q\ -\tau\rightarrow\ P' \parallel Q'$ [PAR3]
$\vdash P\ -u\rightarrow\ Q \land ((u = \tau) \lor (u = \texttt{label}\ l) \land l \notin L \land \texttt{COMPL}\ l \notin L) \Rightarrow$
$\nu\ L\ P\ -u\rightarrow\ \nu\ L\ Q$ [RESTR]
$\vdash P\ -u\rightarrow\ Q \Rightarrow \texttt{relab}\ P\ rf\ -\texttt{relabel}\ rf\ u\rightarrow\ \texttt{relab}\ Q\ rf$ [RELABELING]
$\vdash \texttt{CCS\_Subst}\ P\ (\texttt{rec}\ A\ P)\ A\ -u\rightarrow\ P' \Rightarrow \texttt{rec}\ A\ P\ -u\rightarrow\ P'$ [REC]

The rule [REC] (Recursion) says that if we substitute all appearances of variable A in P to (**rec** A P) and the resulting process has a transition to P' with action u, then (**rec** A P) has the same transition. From the HOL viewpoint, these SOS rules are *inductive definitions* on the 3-ary relation TRANS of type "$(\alpha,\ \beta)$ CCS $\rightarrow\beta$ Action $\rightarrow(\alpha,\ \beta)$ CCS $\rightarrow$bool", generated by HOL4's Hol_reln function (inductive relation package).

A useful function that we have defined, exploiting the interplay between HOL4 and Standard ML (and following an idea by Nesi [11]) is a complex Standard ML function taking a CCS process and returning a theorem indicating all its direct transitions. For instance, we know that the process $(a.0 \mid \bar{a}.0)$ has three possible transitions: $(a.0 \mid \bar{a}.0) \xrightarrow{a} (0 \mid \bar{a}.0)$, $(a.0 \mid \bar{a}.0) \xrightarrow{\bar{a}} (a.0 \mid 0)$ and $(a.0 \mid \bar{a}.0) \xrightarrow{\tau} (0 \mid 0)$. To completely describe all possible transitions of a process, if done manually, the following facts should be proved:

1. the correctness for each of the transitions;
2. the non-existence of other transitions.

For large processes it may be surprisingly hard to manually prove the non-existence of transitions. Hence the usefulness of appealing to the new function `CCS_TRANS_CONV`. For instance this function is called on the process $(a.0 \mid \bar{a}.0)$ thus:

```
> CCS_TRANS_CONV ``par (prefix (label (name "a")) nil)
                       (prefix (label (coname "a")) nil)``
```

This returns the following theorem, indeed describing all immediate transitions of the process:

$$\begin{array}{l} \vdash \texttt{In}\ \text{“a”}\texttt{..nil} \parallel \texttt{Out}\ \text{“a”}\texttt{..nil}\ -u\rightarrow\ E \iff \\ \quad ((u\ \texttt{=}\ \texttt{In}\ \text{“a”})\ \land\ (E\ \texttt{=}\ \texttt{nil} \parallel \texttt{Out}\ \text{“a”}\texttt{..nil})\ \lor \\ \quad\ (u\ \texttt{=}\ \texttt{Out}\ \text{“a”})\ \land\ (E\ \texttt{=}\ \texttt{In}\ \text{“a”}\texttt{..nil} \parallel \texttt{nil}))\ \lor \\ \quad (u\ \texttt{=}\ \tau)\ \land\ (E\ \texttt{=}\ \texttt{nil} \parallel \texttt{nil}) \end{array}$$

5.2 Bisimulation and Bisimilarity

For the definition of (weak) bisimilarity we first need weak transitions (abstracting from τ-moves). We define a (possibly empty) sequence of τ-transitions as a new relation called `EPS` ($\overset{\epsilon}{\Rightarrow}$), which is the RTC (reflexive transitive closure, *) of the single-step τ-transition:

$$\texttt{EPS = } (\lambda E\ E'.\ E\ -\tau\rightarrow\ E')^* \qquad \texttt{[EPS\_def]}$$

Then we can define a weak transition as an ordinary transition wrapped by two ϵ-transitions:

$$E\ =u\Rightarrow\ E' \iff \exists E_1\ E_2.\ E \overset{\epsilon}{\Rightarrow} E_1 \land E_1\ -u\rightarrow\ E_2 \land E_2 \overset{\epsilon}{\Rightarrow} E' \qquad \texttt{[WEAK\_TRANS]}$$

For the definition of bisimilarity and the associated coinduction principle, we have taken advantage of HOL's coinductive relation package (`Hol_coreln` [2]), a new tools since its Kananaskis-11 release (March 3, 2017).[2] This essentially amounts to defining bisimilarity as the greatest fixed-point of the appropriate functional on relations. Precisely we call the `Hol_coreln` command as follows. It returns 3 theorems; we report below two of them, `WB_coind` and `WB_cases`, which express the coinduction proof method for bisimilarity (any bisimulation is contained in bisimilarity) and the fixed-point property of bisimilarity (bisimilarity itself is a bisimulation, thus the largest bisimulation); here `WB` is meant to be bisimilarity (recall also that, in HOL, `!` and `?` stand for universal and existential quantifiers):

```
val (WB_rules, WB_coind, WB_cases) = Hol_coreln `
    (!(P :('a, 'b) CCS) (Q :('a, 'b) CCS).
       (!l.
         (!P'. TRANS P (label l) P' ==>
               (?Q'. WEAK_TRANS Q (label l) Q' /\ WB P' Q')) /\
         (!Q'. TRANS Q (label l) Q' ==>
               (?P'. WEAK_TRANS P (label l) P' /\ WB P' Q'))) /\
       (!P'. TRANS P tau P' ==> (?Q'. EPS Q Q' /\ WB P' Q')) /\
       (!Q'. TRANS Q tau Q' ==> (?P'. EPS P P' /\ WB P' Q'))
      ==> WB P Q)`;
```

[2]https://hol-theorem-prover.org/kananaskis-11.release.html#new-tools

1.
$$\begin{array}{l}
\vdash \forall WB'.\\
\qquad (\forall a_0\ a_1.\\
\qquad\qquad WB'\ a_0\ a_1 \Rightarrow\\
\qquad\qquad (\forall l.\\
\qquad\qquad\qquad (\forall P'.\\
\qquad\qquad\qquad\qquad a_0\ -\texttt{label}\ l\rightarrow\ P' \Rightarrow\\
\qquad\qquad\qquad\qquad \exists Q'.\ a_1\ =\texttt{label}\ l\Rightarrow\ Q' \land WB'\ P'\ Q') \land\\
\qquad\qquad\qquad \forall Q'.\\
\qquad\qquad\qquad\qquad a_1\ -\texttt{label}\ l\rightarrow\ Q' \Rightarrow\\
\qquad\qquad\qquad\qquad \exists P'.\ a_0\ =\texttt{label}\ l\Rightarrow\ P' \land WB'\ P'\ Q') \land\\
\qquad\qquad (\forall P'.\ a_0\ -\tau\rightarrow\ P' \Rightarrow \exists Q'.\ a_1 \overset{\epsilon}{\Rightarrow} Q' \land WB'\ P'\ Q') \land\\
\qquad\qquad \forall Q'.\ a_1\ -\tau\rightarrow\ Q' \Rightarrow \exists P'.\ a_0 \overset{\epsilon}{\Rightarrow} P' \land WB'\ P'\ Q') \Rightarrow\\
\qquad \forall a_0\ a_1.\ WB'\ a_0\ a_1 \Rightarrow \texttt{WB}\ a_0\ a_1
\end{array}$$
[WB_coind]

2.
$$\begin{array}{l}
\vdash \forall a_0\ a_1.\\
\qquad \texttt{WB}\ a_0\ a_1 \iff\\
\qquad (\forall l.\\
\qquad\qquad (\forall P'.\\
\qquad\qquad\qquad a_0\ -\texttt{label}\ l\rightarrow\ P' \Rightarrow\\
\qquad\qquad\qquad \exists Q'.\ a_1\ =\texttt{label}\ l\Rightarrow\ Q' \land \texttt{WB}\ P'\ Q') \land\\
\qquad\qquad \forall Q'.\\
\qquad\qquad\qquad a_1\ -\texttt{label}\ l\rightarrow\ Q' \Rightarrow\\
\qquad\qquad\qquad \exists P'.\ a_0\ =\texttt{label}\ l\Rightarrow\ P' \land \texttt{WB}\ P'\ Q') \land\\
\qquad (\forall P'.\ a_0\ -\tau\rightarrow\ P' \Rightarrow \exists Q'.\ a_1 \overset{\epsilon}{\Rightarrow} Q' \land \texttt{WB}\ P'\ Q') \land\\
\qquad \forall Q'.\ a_1\ -\tau\rightarrow\ Q' \Rightarrow \exists P'.\ a_0 \overset{\epsilon}{\Rightarrow} P' \land \texttt{WB}\ P'\ Q'
\end{array}$$
[WB_cases]

Without the HOL4 coinduction package, bisimilarity would have to be defined following Definition 2.1; then other properties of bisimilarity, such as the fixed-point property in `WB_cases`, would have to be derived manually (which is hard to do; indeed it was one of the main results in Nesi's formalisation work in HOL88 [11]).

5.3 Coarsest (pre)congruence contained in $\approx$ ($\succeq_{\text{bis}}$)

Bisimilarity ($\approx$) is not congruence, since not preserved by the sum operator. For this reason rooted bisimilarity has been introduced (Definition 2.2). In this subsection we discuss two proofs of the result stating that rooted bisimilarity is the coarsest congruence contained in bisimilarity.

The 'coarsest congruence contained in bisimilarity' is the context closure, below called *weak bisimilarity congruence* and formalised thus:

$$\texttt{WEAK\_CONGR = CC WEAK\_EQUIV} \qquad \texttt{[WEAK\_CONGR]}$$
$$\texttt{CC}\ R\ \texttt{=}\ (\lambda g\ h.\ \forall c.\ \texttt{CONTEXT}\ c \Rightarrow R\ (c\ g)\ (c\ h)) \qquad \texttt{[CC\_def]}$$

Given the central role of the sum operator, we also consider the closure of bisimilarity under such operator, below called *sum equivalence* and written $\approx^{+}$:

$$\texttt{SUM\_EQUIV = }(\lambda p\ q.\ \forall r.\ p\ \texttt{+}\ r \approx q\ \texttt{+}\ r) \qquad \texttt{[SUM\_EQUIV]}$$

Rooted bisimilarity, being a congruence contained in bisimilarity, is contained in weak bisimilarity congruence that, in turn, trivially is contained in sum equivalence. Thus the crux is proving that sum equivalence implies rooted bisimilarity:

$$\forall p\, q.\ (\forall r.\ p+r \approx q+r) \Rightarrow p \approx^{c} q \tag{1}$$

The standard argument [10] requires that p and q do not make use of all possible labels. Formalising such an argument requires however a detailed treatment on free and bound names of CCS processes (with the restriction operator being a binder). However, the proof of (1) can be carried out just assuming that the initial weak transitions (those directly emanating from p and q) do not use all possible visible labels. We have formalised this property and called it the *free action* property:

$\texttt{free\_action}\ p \iff \exists a.\ \forall p'.\ \neg(p\ \texttt{=label}\ a\!\Rightarrow\ p')$ [free_action_def]

With this property, the formalisation of (1) says:

$\vdash \texttt{free\_action}\ p \wedge \texttt{free\_action}\ q \Rightarrow (\forall r.\ p + r \approx q + r) \Rightarrow p \approx^c q$

For contraction and rooted contraction, the situation and proof steps are the same, and the corresponding HOL4 theorem says:

$\vdash \texttt{free\_action}\ p \wedge \texttt{free\_action}\ q \Rightarrow (\forall r.\ p + r \succeq_{bis} q + r) \Rightarrow p \succeq_{bis}^c q$

The above proof follows Milner [10], and requires that no process uses all available names. We have also formalised a different proof, by van Glabbeek [9], that does not require additional assumptions. The proof by van Glabbeek uses transfinite induction to obtain a sequence of processes that are all pairwise non-bisimilar. We have formalised van Glabbeek's proof; however, as the typed logic implemented in various HOL systems (including Isabelle/HOL) is not strong enough to define a type for all possible ordinal values [13], we have replaced transfinite induction with plain induction; as a consequence, the final result is about a restricted class of processes (which we have taken to be the finite-state processes). We omit the details for lack of space.

5.4 Context, guardedness and (pre)congruence

We have chosen to use λ-expressions (typed $CCS \to CCS$) to represent (multi-hole) contexts. The definition is inductive:

$$\begin{array}{l}
\vdash \texttt{CONTEXT}\ (\lambda t.\ t) \wedge (\forall p.\ \texttt{CONTEXT}\ (\lambda t.\ p)) \wedge \\
\quad (\forall a\ e.\ \texttt{CONTEXT}\ e \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ a..e\ t)) \wedge \\
\quad (\forall e_1\ e_2.\ \texttt{CONTEXT}\ e_1 \wedge \texttt{CONTEXT}\ e_2 \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ e_1\ t + e_2\ t)) \wedge \\
\quad (\forall e_1\ e_2.\ \texttt{CONTEXT}\ e_1 \wedge \texttt{CONTEXT}\ e_2 \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ e_1\ t \parallel e_2\ t)) \wedge \\
\quad (\forall L\ e.\ \texttt{CONTEXT}\ e \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ \nu\ L\ (e\ t))) \wedge \\
\quad \forall rf\ e.\ \texttt{CONTEXT}\ e \Rightarrow \texttt{CONTEXT}\ (\lambda t.\ \texttt{relab}\ (e\ t)\ rf) \qquad \texttt{[CONTEXT\_rules]}
\end{array}$$

We can then define ‘precongruence’ and ‘congruence’ thus:

$$\begin{array}{l}
\texttt{precongruence}\ R \iff \\
\forall x\ y\ ctx.\ \texttt{CONTEXT}\ ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y) \wedge \texttt{PreOrder}\ R \\
\texttt{congruence}\ R \iff \\
\forall x\ y\ ctx. \\
\quad \texttt{CONTEXT}\ ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y) \wedge \texttt{equivalence}\ R
\end{array}$$

A *weakly guarded* (WG) context is a context in which each hole is underneath a prefix:

$$\begin{array}{l}
\vdash (\forall p.\ \texttt{WG}\ (\lambda t.\ p)) \wedge (\forall a\ e.\ \texttt{CONTEXT}\ e \Rightarrow \texttt{WG}\ (\lambda t.\ a..e\ t)) \wedge \\
\quad (\forall e_1\ e_2.\ \texttt{WG}\ e_1 \wedge \texttt{WG}\ e_2 \Rightarrow \texttt{WG}\ (\lambda t.\ e_1\ t + e_2\ t)) \wedge \\
\quad (\forall e_1\ e_2.\ \texttt{WG}\ e_1 \wedge \texttt{WG}\ e_2 \Rightarrow \texttt{WG}\ (\lambda t.\ e_1\ t \parallel e_2\ t)) \wedge \\
\quad (\forall L\ e.\ \texttt{WG}\ e \Rightarrow \texttt{WG}\ (\lambda t.\ \nu\ L\ (e\ t))) \wedge \\
\quad \forall rf\ e.\ \texttt{WG}\ e \Rightarrow \texttt{WG}\ (\lambda t.\ \texttt{relab}\ (e\ t)\ rf) \qquad \texttt{[WG\_rules]}
\end{array}$$

A *strongly guarded* (SG) context is a context in which each hole is underneath a visible prefix:

$$\begin{array}{l}
\vdash (\forall p.\ \texttt{SG}\ (\lambda t.\ p)) \land \\
\quad (\forall l\ e.\ \texttt{CONTEXT}\ e \Rightarrow \texttt{SG}\ (\lambda t.\ \texttt{label}\ l\texttt{..}e\ t)) \land \\
\quad (\forall a\ e.\ \texttt{SG}\ e \Rightarrow \texttt{SG}\ (\lambda t.\ a\texttt{..}e\ t)) \land \\
\quad (\forall e_1\ e_2.\ \texttt{SG}\ e_1 \land \texttt{SG}\ e_2 \Rightarrow \texttt{SG}\ (\lambda t.\ e_1\ t\ \texttt{+}\ e_2\ t)) \land \\
\quad (\forall e_1\ e_2.\ \texttt{SG}\ e_1 \land \texttt{SG}\ e_2 \Rightarrow \texttt{SG}\ (\lambda t.\ e_1\ t \parallel e_2\ t)) \land \\
\quad (\forall L\ e.\ \texttt{SG}\ e \Rightarrow \texttt{SG}\ (\lambda t.\ \nu\ L\ (e\ t))) \land \\
\quad \forall rf\ e.\ \texttt{SG}\ e \Rightarrow \texttt{SG}\ (\lambda t.\ \texttt{relab}\ (e\ t)\ rf) \qquad \texttt{[SG\_rules]}
\end{array}$$

And a *sequential* (SEQ) context is a context in which each hole is under prefix or sums:

$$\begin{array}{l}
\vdash \texttt{SEQ}\ (\lambda t.\ t) \land (\forall p.\ \texttt{SEQ}\ (\lambda t.\ p)) \land \\
\quad (\forall a\ e.\ \texttt{SEQ}\ e \Rightarrow \texttt{SEQ}\ (\lambda t.\ a\texttt{..}e\ t)) \land \\
\quad \forall e_1\ e_2.\ \texttt{SEQ}\ e_1 \land \texttt{SEQ}\ e_2 \Rightarrow \texttt{SEQ}\ (\lambda t.\ e_1\ t\ \texttt{+}\ e_2\ t) \qquad \texttt{[SEQ\_rules]}
\end{array}$$

In the same manner, we can also define all variants of above concepts with only weakly-guarded sums (GCONTEXT, WGS and GSEQ). Several lemmas have then to be proved for their relationships among these kinds of contexts and properties about their compositions. These proofs are usually tedious but long, due to multiple levels of inductions on the structure of the contexts.

5.5 Unique solution of contractions

A delicate point in the formalisation of the results about unique solution of contractions are the proof of lemma 4.3 and alike lemmas; in particular, there is an induction on the length of weak transitions. For this, rather than introducing a refined form of weak transition relation enriched with its length, we found it more elegant to work with traces (a motivation for this is to set the ground for extensions of this formalisation work to trace equivalence in place of bisimilarity).

A trace is represented by the initial and final processes, plus a list of actions so performed. For this, we first define the Reflexive Transitive Closure with a List (LRTC): given a labelled transition relation R, then LRTC R builds the trace relation derived from R:

$$\begin{array}{l}
\vdash \texttt{LRTC}\ R\ a\ l\ b \iff \\
\quad \forall P. \\
\qquad (\forall x.\ P\ x\ \texttt{[]}\ x) \land \\
\qquad (\forall x\ h\ y\ t\ z.\ R\ x\ h\ y \land P\ y\ t\ z \Rightarrow P\ x\ (h\texttt{::}t)\ z) \Rightarrow \\
\qquad P\ a\ l\ b \qquad \texttt{[LRTC\_DEF]}
\end{array}$$

The traces for a CCS processes are obtained by combining LRTC with the CCS transition relation TRANS:

$$\texttt{TRACE = LRTC TRANS} \qquad \texttt{[TRACE\_def]}$$

If there is at most one visible action (i.e., a label) in the list of actions of a trace, then the trace also represents a weak transition. Here we distinguish two cases: no label and unique label. The definition of 'no label' in an action list is easy (below MEM tests if a given element is a member of a list):

$$\vdash \texttt{NO\_LABEL}\ L \iff \neg\exists l.\ \texttt{MEM}\ (\texttt{label}\ l)\ L \qquad \texttt{[NO\_LABEL\_def]}$$

The definition below of 'unique label' avoids counting or filtering in a list; it says that a label is unique in an action list iff it there is no other label in the remainder part of the list:

$$\begin{array}{l}
\vdash \texttt{UNIQUE\_LABEL}\ u\ L \iff \\
\quad \exists L_1\ L_2.\ (L_1 \mathbin{+\!\!+} [u] \mathbin{+\!\!+} L_2 = L) \land \texttt{NO\_LABEL}\ L_1 \land \texttt{NO\_LABEL}\ L_2\ \ [\texttt{UNIQUE\_LABEL\_def}]
\end{array}$$

The final relationship between traces and weak transitions is stated and proved in the following theorem (where the variable *acts* stands for a list of actions); it says that a weak transition $P \overset{u}{\Rightarrow} P'$ is a trace with a non-empty action list, i.e., either without any (visible) label if $u = \tau$, or else if $u \neq \tau$ then u is the unique label in the list:

$$\begin{array}{l}
\vdash P \ {=}u{\Rightarrow}\ P' \iff \\
\quad \exists \mathit{acts}. \\
\qquad \texttt{TRACE}\ P\ \mathit{acts}\ P' \land \neg\texttt{NULL}\ \mathit{acts} \land \\
\qquad \textbf{if}\ u = \tau\ \textbf{then}\ \texttt{NO\_LABEL}\ \mathit{acts}\ \textbf{else}\ \texttt{UNIQUE\_LABEL}\ u\ \mathit{acts}\ \ [\texttt{WEAK\_TRANS\_AND\_TRACE}]
\end{array}$$

We show the formalized statement of Lemma 3.9:

$$\begin{array}{l}
\vdash (\exists E.\ \texttt{WGS}\ E \land P \succeq_{bis} E\ P \land Q \succeq_{bis} E\ Q) \Rightarrow \\
\quad \forall C. \\
\qquad \texttt{GCONTEXT}\ C \Rightarrow \\
\qquad (\forall l\ R. \\
\qquad\quad C\ P\ {=}\texttt{label}\ l{\Rightarrow}\ R \Rightarrow \\
\qquad\quad \exists C'. \\
\qquad\qquad \texttt{GCONTEXT}\ C' \land R \succeq_{bis} C'\ P \land \\
\qquad\qquad (\texttt{WEAK\_EQUIV} \circ_r (\lambda x\ y.\ x\ {=}\texttt{label}\ l{\Rightarrow}\ y))\ (C\ Q) \\
\qquad\qquad\ \ (C'\ Q)) \land \\
\qquad \forall R. \\
\qquad\quad C\ P\ {=}\tau{\Rightarrow}\ R \Rightarrow \\
\qquad\quad \exists C'. \\
\qquad\qquad \texttt{GCONTEXT}\ C' \land R \succeq_{bis} C'\ P \land \\
\qquad\qquad (\texttt{WEAK\_EQUIV} \circ_r \texttt{EPS})\ (C\ Q)\ (C'\ Q)
\end{array}$$

Traces are introduced in the proof of above lemma via the following 'unfolding' lemma:

$$\begin{array}{l}
\vdash \texttt{GCONTEXT}\ C \land \texttt{WGS}\ E \land \texttt{TRACE}\ ((C \circ \texttt{FUNPOW}\ E\ n)\ P)\ xs\ P' \land \\
\quad \texttt{LENGTH}\ xs \le n \Rightarrow \\
\quad \exists C'. \\
\qquad \texttt{GCONTEXT}\ C' \land (P' = C'\ P) \land \\
\qquad \forall Q.\ \texttt{TRACE}\ ((C \circ \texttt{FUNPOW}\ E\ n)\ Q)\ xs\ (C'\ Q)
\end{array}$$

It roughly says that, for any context C and weakly guarded expression E, if $C[E^n[P]] \overset{xs}{\Longrightarrow} P'$ and the length of actions $xs \leqslant n$, then P has the form of $C'[P]$ (meaning that P is not touched during the transitions). Traces are necessary to reason about the length of intermediate actions in weak transitions.

With all above work, finally we can formally prove Theorem 3.10 (here `WGS` stands for 'weakly guarded context with weakly-guarded sums'):

$$\vdash \texttt{WGS}\ E \Rightarrow \forall P\ Q.\ P \succeq_{bis} E\ P \land Q \succeq_{bis} E\ Q \Rightarrow P \approx Q$$

5.6 Unique solution of rooted contractions

The formal proof of "unique solution of rooted contractions theorem" (Theorem 4.4) has the same initial proof steps as Theorem 3.10; it then requires a few more steps to handle rooted bisimilarity in the conclusion. Overall the two proofs are very similar, mostly because the only property we need from (rooted) contraction is its precongruence. Below is the formally verified version of Theorem 4.4 (having proved the precongruence of rooted contraction, we can use weakly-guarded expressions, without constraints on sums; that is, `WG` in place of `WGS`):

$\vdash \texttt{WG}\ E \Rightarrow \forall P\ Q.\ P \succeq^c_{bis}\ E\ P \land Q \succeq^c_{bis}\ E\ Q \Rightarrow P \approx^c Q$

Having removed the constraints on sums, the result is similar to Milner's original 'unique solution of equations' theorem for *strong* bisimilarity ($\sim$) — the same weakly guarded condition (WG) is required:

$\vdash \texttt{WG}\ E \Rightarrow \forall P\ Q.\ P \sim E\ P \land Q \sim E\ Q \Rightarrow P \sim Q$

In contrast, Milner's 'unique solution of equations' theorem for rooted bisimilarity ($\approx^{\mathrm{c}}$) has more severe constraints:

$\vdash \texttt{SG}\ E \land \texttt{SEQ}\ E \Rightarrow \forall P\ Q.\ P \approx^c E\ P \land Q \approx^c E\ Q \Rightarrow P \approx^c Q$

6 Related formalisation work

The closest of our work is Monica Nesi's [11, 12]. She has made the first formalization of CCS, using early versions of the HOL theorem prover. Her main focus is on decision procedures (as ML functions) for automatically proving bisimilarities between CCS processes. Her work has been a basis for ours. However, the differences are substantial, both because we have used the most recent version of HOL, and because of the spectrum of the theory considered.

Bengtson, Parrow and Weber have made a substantial formalisation work on CCS, π-calculi and ψ-calculi using Isabelle/HOL and its nominal datatype package, with main focus on the handling of name binders [4, 5, 6]. Other formalisations in this area include the work of Solange Coupet-Grimal[3] in Coq and Chaudhuri et al. [7] in Abella. The latter focuses on 'bisimulation up-to' techniques for strong bisimilarity for CCS and π-calculus.

7 Conclusions and future work

We have highlighted a formalisation of the theory of CCS in the HOL4 theorem prover (for lack of space we have not discussed the formalisation of some basic algebraic theory, of the basic properties of the expansion preorder, and of a few versions of 'bisimulation up to' techniques). The formalisation has focused on the theory of unique solution of equations and contractions. It has also allowed us to further develop the theory, notably the basic properties of rooted contraction, and the unique solution theorem for it with respect to rooted bisimilarity. The formalisation brings up and exploits similarities between results and proofs for different equivalences and preorders. We think that the statements in the formalisation are easy to read and understand, as they remain very close to the ordinary statements of the theory of CCS. Moreover, the logic foundations of HOL makes the whole work (or individual parts) easily portable to other theorem provers.

As future work, it would be worth extending the current work to multiple equations and contractions; and considering other equivalences and preorders, notably trace relations. On another line, one could examine the formalisation of a different approach to unique solutions, by Durier et al. [8], in which the use of contraction is replaced by semantic conditions on process transitions such as divergence.

[3] `https://github.com/coq-contribs/ccs`

Acknowledgements We have benefitted from suggestions and comments from several people from the HOL community, including Robert Beers, Jeremy Dawson, Ramana Kumar, Michael Norrish, Konrad Slind, and Thomas Tuerk.

References

[1] (2016): *The HOL System LOGIC*, pp. 1–45. Available at `http://sourceforge.net/projects/hol/files/hol/kananaskis-11/kananaskis-11-logic.pdf/download`.

[2] (2017): *The HOL System DESCRIPTION*, pp. 1–349. Available at `http://sourceforge.net/projects/hol/files/hol/kananaskis-11/kananaskis-11-description.pdf/download`.

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

[4] Jesper Bengtson (2010): *Formalising process calculi*. Ph.D. thesis, Acta Universitatis Upsaliensis.

[5] Jesper Bengtson & Joachim Parrow (2007): *A completeness proof for bisimulation in the pi-calculus using isabelle*. *Electronic Notes in Theoretical Computer Science* 192(1), pp. 61–75.

[6] Jesper Bengtson & Joachim Parrow (2007): *Formalising the π-calculus using nominal logic*. In: *International Conference on Foundations of Software Science and Computational Structures*, Springer, pp. 63–77.

[7] Kaustuv Chaudhuri, Matteo Cimini & Dale Miller (2015): *A lightweight formalization of the metatheory of bisimulation-up-to*. In: *Proceedings of the 2015 Conference on Certified Programs and Proofs*, ACM, pp. 157–166.

[8] Adrien Durier, Daniel Hirschkoff & Davide Sangiorgi (2017): *Divergence and Unique Solution of Equations*. In: *Proc. CONCUR 2017*, *LIPIcs* 85, Schloss Dagstuhl, pp. 11:1–11:16.

[9] R. J. van Glabbeek (2005): *A characterisation of weak bisimulation congruence*. *Lecture notes in computer science* 3838, pp. 26–39.

[10] Robin Milner (1989): *Communication and concurrency*. Prentice-Hall.

[11] Monica Nesi (1992): *A formalization of the process algebra CCS in high order logic*. Technical Report, University of Cambridge, Computer Laboratory.

[12] Monica Nesi (1999): *Formalising a Value-Passing Calculus in HOL*. *Formal Aspects of Computing* 11(2), pp. 160–199.

[13] Michael Norrish & Brian Huffman (2013): *Ordinals in HOL: Transfinite arithmetic up to (and beyond) ω_1*. In: *International Conference on Interactive Theorem Proving*, Springer, pp. 133–146.

[14] A. W. Roscoe (1998): *The theory and practice of concurrency*. Prentice Hall. Available at `http://www.cs.ox.ac.uk/people/bill.roscoe/publications/68b.pdf`.

[15] Davide Sangiorgi (2015): *Equations, contractions, and unique solutions*. In: *ACM SIGPLAN Notices*, 50, ACM, pp. 421–432.

[16] Konrad Slind & Michael Norrish (2008): *A brief overview of HOL4*. In: *International Conference on Theorem Proving in Higher Order Logics*, Springer, pp. 28–32.

[17] Chun Tian (2017): *A Formalization of Unique Solutions of Equations in Process Algebra*. Master's thesis, AlmaDigital, Bologna. Available at `http://amslaurea.unibo.it/14798/`.