Unique Solutions of Contractions, CCS, and their HOL Formalisation[⋆]

Chun Tian[a], Davide Sangiorgi[b]

[a] *Università di Trento and Fondazione Bruno Kessler, Italy*
[b] *Università di Bologna and INRIA, Italy*

Abstract

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 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 summation, which must be weakly-guarded, by moving to *rooted contraction*. We prove that rooted contraction is indeed the coarsest precongruence contained in the contraction preorder, in the same way as rooted bisimilarity is so for weak bisimilarity.

Keywords: process calculi, theorem provers, coinduction, unique solution of equations, congruence

1. Introduction

A prominent proof method for bisimulation, put forward by Robin Milner and widely used in his landmark CCS book [2], 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, 4, 5].

In Milner's versions of the unique-solution theorems for weak equivalences, the method has severe syntactical limitations: the equations must be *(strongly) guarded and sequential.* That is, the variables of the equations may only be used underneath a visible prefix and proceed, in the syntax tree, only by the sum and prefix operators. One way of overcoming such limitations is to replace the equations with special inequations called *contractions* [6, 7]. Contraction is a preorder that, roughly, places some efficiency constraints on processes. The uniqueness of solutions of a system of contractions is defined as with systems of equations: any two solutions must be (componentwise) 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 [7] (in which the issue is bypassed using a more restrictive CCS syntax) is the occurrences of direct sums, due to the failure of the substitutivity of contraction under direct sums.

The main goal of the work described in this paper is a rather comprehensive formalisation of the core theory of CCS in the HOL theorem prover (HOL4), with a focus on the theory of unique solutions of equations and contractions. The formalisation, however, is not confined to the theory of unique solutions, but embraces a significant portion the theory of CCS [2], mostly because the theory of unique solutions relies on a large 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), the basic properties of rooted bisimilarity (the congruence induced by weak bisimilarity, also called observation congruence), and

[⋆]An extended version of the EXPRESS/SOS 2018 workshop paper [1].
Email addresses: `chun.tian@unitn.it` (Chun Tian), `davide.sangiorgi@unibo.it` (Davide Sangiorgi)

their algebraic laws. Further extensions (beyond Nesi [8]) include four versions of "bisimulation up to" techniques (e.g. bisimulation up-to bisimilarity) [2, 9], and the expansion and contraction preorder (two efficiency-like refinements of weak bisimilarity). Concerning rooted bisimilarity, the formalisation includes Hennessy Lemma and Deng Lemma (Lemma 4.1 and 4.2 of [10]), and two long proofs saying that rooted bisimilarity is the coarsest (largest) congruence contained in (weak) bisimilarity: one following Milner's book [2], with the hypothesis that no processes can use up all labels; the other without such hypothesis, essentially formalising van Glabbeek's paper [11]. 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 with its recent developments, including a package for coinductive definitions.

From the view of CCS theory, this formalisation has offered us the possibility of further refining the theory of unique solutions of contractions, as formally proving a previously known result gives us the chance to see *what is really needed* for establishing that result. In particular, the existing theory [7] has 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 contraction-based proof technique, by moving to *rooted contraction*, that is, the coarsest precongruence contained in the contraction preorder. The resulting unique-solution theorem is now valid for *rooted bisimilarity* (hence also for bisimilarity itself), and places no constraints on the occurrences of sums.

Another benefit of the formalisation is that we can take advantage of results about different equivalences and preorders that share similar proof structures. 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 enhancements 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 proof scripts have to be modified. Then the successful termination of the proof gives us a guarantee that the proof is complete and trustworthy, eliminating the risks of overlooking or missing details as in *paper-and-pencil* proofs.

Structure of the paper. Section 2 presents basic background materials on CCS, including its syntax, operational semantics, bisimilarity and rooted bisimilarity. Section 3 discusses equations and contractions. Section 3.4 presents rooted contraction and the related unique-solution result for rooted bisimilarity. Section 4 highlights our formalisation in HOL4. Finally, Section 5 and 6 discuss related work, conclusions, and a few directions for future work.

2. CCS

We assume a possibly infinite set of names $\mathscr{L} = \{a, b, \ldots\}$ forming input and $\overline{\text{output}}$ actions, plus a special invisible action $\tau \notin \mathscr{L}$, and a set of variables $A, B, \ldots$ for defining recursive behaviours. Given a deadlock $\mathbf{0}$, the class of CCS processes is inductively defined from $\mathbf{0}$ by the operators of prefixing, parallel composition, summation (binary choice), restriction, recursion and relabeling:

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

We sometimes omit the trailing $\mathbf{0}$, e.g., writing $a \mid b$ for $a.\mathbf{0} \mid b.\mathbf{0}$. The operational semantics of CCS is then given by means of a Labeled Transition System (LTS), shown in Fig. 1 as Structural Operational Semantics (SOS) rules (the symmetric version of the two rules for parallel composition and the rule for sum are omitted). A CCS expression uses only *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 $\{P' \mid P \xrightarrow{\mu} P' \text{ for some } \mu\}$. We use ℓ to range over visible actions (i.e. inputs or outputs, excluding τ) and α, μ to range over all actions. Some standard notations for transitions: $\overset{\epsilon}{\Longrightarrow}$ is the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\overset{\mu}{\Longrightarrow}$ is $\overset{\epsilon}{\Longrightarrow}\xrightarrow{\mu}\overset{\epsilon}{\Longrightarrow}$ (the composition of the three relations).

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

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

Figure 1: Structural Operational Semantics of CCS

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. We let $\mathcal{R}$, $\mathcal{S}$ range over binary relations, sometimes using infix notation for them; e.g., $P \mathrel{\mathcal{R}} Q$ means that $(P, Q) \in \mathcal{R}$. We use a tilde, as in $\widetilde{P}$, to denote tuples of processes with countably many elements (thus the tuple may also be infinite.) All notations are extended to tuples componentwise, e.g., $\widetilde{P} \mathrel{\mathcal{R}} \widetilde{Q}$ means that $P_i \mathrel{\mathcal{R}} Q_i$, 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 context C and tuples $\widetilde{P}, \widetilde{Q}$ with $P = C[\widetilde{P}], Q = C[\widetilde{Q}]$ and $\widetilde{P} \mathrel{\mathcal{R}} \widetilde{Q}$. We use the 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 here are mainly *weak* ones, in that they abstract from the number of internal steps being performed:

Definition 2.1. *A process relation $\mathcal{R}$ is a* bisimulation *if, whenever $P \mathrel{\mathcal{R}} Q$, for all μ we have:*

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

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

We sometimes call bisimilarity the *weak* one, 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 $Q \xrightarrow{\mu} Q'$. Weak bisimilarity is not preserved by the sum operator (except for guarded sums). For this, Milner introduced *observational congruence*, also called *rooted bisimilarity* [10, 12]:

Definition 2.2. *Two processes P and Q are* rooted bisimilar*, written as $P \approx^{\mathrm{c}} Q$, if for all μ:*

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

Besides reducing the rooted bisimiarity of two processes to the bisimilarities of their immediate derivatives, the above definition also brings up a proof technique for proving rooted bisimiarity results by constructing a bisimulation.

Lemma 2.3 (Rooted bisimilarity by bisimulation)**.** *Given a (weak) bisimulation $\mathcal{R}$, if two processes P and Q satisfies the following properties:*

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

then P and Q are rooted bisimilar, i.e. $P \approx^{\mathrm{c}} Q$.

Theorem 2.4. *$\approx^{\mathrm{c}}$ is a congruence in CCS, and it is the coarsest congruence contained in $\approx$ [11].*

3. Equations and contractions

3.1. Systems of equations

Uniqueness of solutions of equations [2] 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 $R \overset{\text{def}}{=} \texttt{rec}\, A.\ (a.\, A)$, 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, for instance, K and $K \mid b$, for $K \overset{\text{def}}{=} \texttt{rec}\, K.\ (a.\, K)$. (Actually any process capable of continuously performing a **actions** is a solution for $X = a \mid X$.) *ds*

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

- weakly guarded *if, in each E_i, each occurrence of an equation variable is underneath a prefix;*
- (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 of its subexpressions with occurrence of an equation variable, apart from the variable itself, is in forms of prefixes or sums.*

Theorem 3.4 (unique solution of equations, [2])**.** *A system of guarded and sequential equations (without direct sums) $\{X_i = E_i\}_{i\in I}$ has a unique solution for $\approx$.*

To see the need of the sequentiality condition, consider the equation (from [2]) $X = \boldsymbol{\nu} a\, (a.\, X \mid \overline{a})$ where X is guarded but not sequential. Any process 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 a special kind of inequations called *contractions.*

Intuitively, the bisimilarity contraction $\succeq_{\text{bis}}$ is a preorder in which $P \succeq_{\text{bis}} Q$ 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. *A process relation $\mathcal{R}$ is a* (bisimulation) contraction *if, whenever $P \mathrel{\mathcal{R}} Q$,*

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

Bisimilarity contraction*, written as $P \succeq_{\text{bis}} Q$ (P* contracts to *Q), if $P \mathrel{\mathcal{R}} Q$ for some contraction $\mathcal{R}$.*

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* $\succeq_{\mathrm{e}}$ [13, 6]. 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 $P \overset{\mu}{\Longrightarrow} P'$, rather than $P \overset{\widehat{\mu}}{\Longrightarrow} P'$; moreover with contraction the final derivatives are simply required to be bisimilar. An expansion $P \succeq_{\mathrm{e}} Q$ tells us that Q is always at least as efficient as P, whereas the contraction $P \succeq_{\mathrm{bis}} Q$ just says that Q has the possibility of being at least as efficient as P.

Definition 3.6. *A process relation $\mathcal{R}$ is an* expansion *if, whenever we have $P \mathrel{\mathcal{R}} Q$, for all μ*

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

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

As bisimilarity, so the expansion preorder is preserved by all operators but sums.

Example 3.7. *We have $a \not\succeq_{\mathrm{bis}} \tau.\, a$. However, $a + \tau.\, a \succeq_{\mathrm{bis}} a$, as well as its converse, $a \succeq_{\mathrm{bis}} a + \tau.\, a$. Indeed, if $P \approx Q$ then $P \succeq_{\mathrm{bis}} P + Q$. The last two relations do not hold with $\succeq_{\mathrm{e}}$, which explains the strictness of the inclusion $\succeq_{\mathrm{e}} \subset \succeq_{\mathrm{bis}}$.*

Like (weak) bisimilarity and expansion, contraction is preserved by all operators but (direct) 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 3.8. *Given a system of contractions $\{X_i \succeq E_i\}_{i \in I}$, we say that:*

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

The guardedness of contractions follows Def. 3.3 (for equations).

Lemma 3.9. *Suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\succeq_{\mathrm{bis}}$ of a system of weakly-guarded contractions that uses weakly-guarded sums. For any context C that uses weakly-guarded sums, if $C[\widetilde{P}] \overset{\mu}{\Longrightarrow} R$, then there is a context C' that uses weakly-guarded sums such that $R \succeq_{\mathrm{bis}} C'[\widetilde{P}]$ and $C[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow}\approx C'[\widetilde{Q}]$.*[1]

Proof. (sketch from [7]) Let n be the length of the transition $C[\widetilde{P}] \overset{\mu}{\Longrightarrow} R$ (the number of 'strong steps' of which it is composed), and let $C''[\widetilde{P}]$ and $C''[\widetilde{Q}]$ be the processes obtained from $C[\widetilde{P}]$ and $C[\widetilde{Q}]$ by unfolding the definitions of the contractions n times. Thus in C'' each hole is underneath at least n prefixes, and cannot contribute to an action in the first n transitions; moreover all the contexts have only 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 $C[\widetilde{P}] \overset{\mu}{\Longrightarrow} R$, we infer the existence of C' such that $C''[\widetilde{P}] \overset{\widehat{\mu}}{\Longrightarrow} C'[\widetilde{P}] \preceq_{\mathrm{bis}} R$ and $C''[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow} C'[\widetilde{Q}]$. Finally, again applying the definition of $\succeq_{\mathrm{bis}}$ on $C[\widetilde{Q}] \succeq_{\mathrm{bis}} C''[\widetilde{Q}]$, we derive $C[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow}\approx C'[\widetilde{Q}]$. $\square$

[1]There's no typo here: $C[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow}\approx C'[\widetilde{Q}]$ means $\exists \widetilde{R}.\ C[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow} \widetilde{R} \approx C'[\widetilde{Q}]$. Same as in Lemma 3.13.

Theorem 3.10 (unique solution of contractions for $\approx$)**.** *A system of weakly-guarded contractions having only weakly-guarded sums, has a unique solution for $\approx$.*

Proof. (sketch from [7]) Suppose $\widetilde{P}$ and $\widetilde{Q}$ are two such solutions (for $\approx$) and consider the relation

$$\mathcal{R} \overset{\text{def}}{=} \{(R,S) \mid R \approx C[\widetilde{P}], S \approx C[\widetilde{Q}] \text{ for some context } C \text{ (weakly-guarded sum only)}\} \ . \tag{1}$$

We show that $\mathcal{R}$ is a bisimulation. Suppose $R \mathrel{\mathcal{R}} S$ vis the context C, and $R \xrightarrow{\mu} R'$. We have to find S' with $S \overset{\widehat{\mu}}{\Longrightarrow} S'$ and $R' \mathrel{\mathcal{R}} S'$. From $R \approx C[\widetilde{P}]$, we derive $C[\widetilde{P}] \overset{\widehat{\mu}}{\Longrightarrow} R'' \approx R'$ for some R''. By Lemma 3.9, there is C' with $R'' \succeq_{\text{bis}} C'[\widetilde{P}]$ and $C[\widetilde{Q}] \overset{\widehat{\mu}}{\Longrightarrow}\approx C'[\widetilde{Q}]$. Hence, by definition of $\approx$, there is also S' with $S \overset{\widehat{\mu}}{\Longrightarrow} S' \approx C'[\widetilde{Q}]$. This closes the proof, as we have $R' \approx C'[\widetilde{P}]$ and $S' \approx C'[\widetilde{Q}]$. $\square$

3.4. 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 3.11. *Two processes P and Q are in* rooted contraction*, written as $P \succeq^{\text{c}}_{\text{bis}} Q$, if*

1. *$P \xrightarrow{\mu} P'$ implies that there is Q' with $Q \xrightarrow{\mu} Q'$ and $P' \succeq_{\text{bis}} Q'$;*
2. *$Q \xrightarrow{\mu} Q'$ implies that there is P' with $P \overset{\mu}{\Longrightarrow} P'$ and $P' \approx Q'$.*

The precise formulation of this definition was guided by the HOL theorem prover and the following two principles: (1) the definition should not be recursive, along the lines of rooted bisimilarity $\approx^{\text{c}}$ in Def. 2.2; (2) the definition should be built on top of existing *contraction* relation $\succeq_{\text{bis}}$ (because of its completeness). A few other candidates were quickly tested and rejected, e.g., because of precongruence issue. The proof of the precongruence result below is along the lines of the analogous result for rooted bisimilarity with respect to bisimilarity.

Theorem 3.12. *$\succeq^{\text{c}}_{\text{bis}}$ is a precongruence in CCS, and it is the coarsest precongruence contained in $\succeq_{\text{bis}}$.*

For a system of rooted contractions, the meaning of "solution for $\succeq^{\text{c}}_{\text{bis}}$" and of *a unique solution for* $\approx^{\text{c}}$ is the expected one — just replace in Def. 3.8 the preorder $\succeq_{\text{bis}}$ with $\succeq^{\text{c}}_{\text{bis}}$, and the equivalence $\approx$ with $\approx^{\text{c}}$ (note that with $\approx^{\text{c}}$ one considers solutions with respect to $\succeq^{\text{c}}_{\text{bis}}$). For this new relation, 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 almost identical, because all properties of $\succeq^{\text{c}}_{\text{bis}}$ needed in this proof is its precongruence, which is indeed true on unrestricted contexts including direct sums:

Lemma 3.13. *Suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\succeq^{\text{c}}_{\text{bis}}$ of a system of weakly-guarded contractions. For any context C, if $C[\widetilde{P}] \overset{\mu}{\Longrightarrow} R$, then there is a context C' such that $R \succeq_{\text{bis}} C'[\widetilde{P}]$ and $C[\widetilde{Q}] \overset{\mu}{\Longrightarrow}\approx C'[\widetilde{Q}]$.*

The proof of Lemma 3.13 is exactly the same as of Lemma 3.9, because all we need from $\succeq_{\text{bis}}$ and $\succeq^{\text{c}}_{\text{bis}}$ is their precongruences (in the sense of different definitions of contexts). Also notice that, the conclusion of Lemma 3.13 still uses $\succeq_{\text{bis}}$, because $\succeq^{\text{c}}_{\text{bis}}$ does not hold.

Theorem 3.14 (Unique solution of contractions for $\approx^{\text{c}}$)**.** *A system of weakly-guarded contractions has a unique solution for $\approx^{\text{c}}$.*

Proof. Suppose $\widetilde{P}$ and $\widetilde{Q}$ are two such solutions: $\widetilde{P} \succeq^{\text{c}}_{\text{bis}} C[\widetilde{P}]$ and $\widetilde{Q} \succeq^{\text{c}}_{\text{bis}} C[\widetilde{Q}]$ (thus also $\widetilde{P} \approx C[\widetilde{P}]$ and $\widetilde{Q} \approx C[\widetilde{Q}]$). We first follow the same steps as in the proof of Theorem 3.10 to show that the following relation is bisimulation (using Lemma 3.13 in place of Lemma 3.9):

$$\mathcal{R} \overset{\text{def}}{=} \{(R,S) \mid R \approx C[\widetilde{P}], S \approx C[\widetilde{Q}] \text{ for some context } C\} \ . \tag{2}$$

The property that $\mathcal{R}$ is a bisimulation proves $\widetilde{P}$ and $\widetilde{Q}$ bisimilar, i.e., $\widetilde{P} \approx \widetilde{Q}$. To further prove $\widetilde{P} \approx^{c} \widetilde{Q}$, we appeal to Lemma 2.3. Thus it remains to show that, for any two process P and Q and action μ, if $P \xrightarrow{\mu} P'$ then there is Q' such that $Q \overset{\mu}{\Longrightarrow} Q'$ and $P' \mathcal{R} Q'$ (and the converse direction). The prove for this part is routine, using Lemma 3.13 of [2], reported as Lemma 4.14 in Section 4.9 (the lemma is also needed in Milner's proof of "unique solution of equations for $\sim$"). □

4. The formalisation

We highlight here a formalisation of CCS in the HOL theorem prover (HOL4) [14], including the new concepts and theorems proposed in the first half of this paper. The whole formalisation (apart from minor fixes and extensions in this paper) is described in [15], and the proof scripts are in HOL4 official examples[2]. The current work consists of about 20,000 lines of proof scripts.

Higher Order Logic (HOL) [16] traces its roots back to LCF [17, 18] by Robin Milner and others since 1972, it is a variant of Church's simple theory of types (STT) [19], plus a higher order version of Hilbert's choice operator ε, Axiom of Infinity, and Rank-1 (prenex) polymorphism. HOL4 has 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 logical foundations than most other theorem provers. As a consequence, formal theories built in HOL is easily convincible and can also be easily ported to other proof systems, sometimes automatically [20].

HOL4 is written in Standard ML, a single programming language which plays three different roles:

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.

Moreover, using the same language HOL4 users can write complex automatic verification tools by calling HOL's theorem proving facilities. (The formal proofs of theorems in CCS theory are mostly done by an *interactive process* closely following their informal proofs, with minimal automatic proof searching.)

In this formalisation we consider only single-variable equations/contractions. This considerably simplifies the required proofs in HOL, and enhances the readability of proof scripts without loss of generality. For the pencil-and-paper proofs, the multi-variable case is just a routine adaptation, while the **theorem-prover** formalisation is not (this is also discussed in Section 6). *ds*

4.1. CCS processes and their transitions

In our CCS formalisation, the type "β `Label`" (`'b` or β is the type variable for actions) accounts for visible actions, divided into input and output actions, defined by HOL's Datatype package:

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

The type "β `Action`" is the union of all visible actions, plus invisible action τ (now based on HOL's `option` theory). The cardinality of "β `Action`" (and therefore of all CCS types built on top of it) depends on the choice (or *type-instantiation*) of type variable β.

The type "`(`α`,` β`) CCS`", accounting for the CCS syntax[3], is then defined inductively: (`'a` or α is the type variable for recursion variables, "β `Relabeling`" is the type of all relabeling functions, `‘` is for backquotes of HOL terms):

[2]`https://github.com/HOL-Theorem-Prover/HOL/tree/master/examples/CCS`

[3]The order of type variables α and β is arbitrary. Our choice is aligned with other CCS literals. CCS(h, k) is the CCS subcalculus that can use at most h constants and k actions [21]. Thus, to formalize theorems on such a calculus, the needed CCS type can be retrieved by instantiating the type variables α and β in "`(`α`,` β`) CCS`" with types having the corresponding cardinalities h and k. Monica Nesi goes farther, by adding another type variable γ for value-passing CCS [22].

```
val _ = Datatype ‘CCS = nil
                      | var 'a
                      | 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's powerful pretty printer, to represent CCS processes in more readable forms (c.f. the column "HOL (abbrev.)" of Table 1, which summarizes the main syntactic notations of 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`..`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`	ν L P
recursion	`rec` $A.P$	`rec A P`	`rec` A P
relabeling	$P\ [rf]$	`relab P rf`	`relab` P rf
constant	A	`var A`	`var` A
invisible action	τ	`tau`	τ
input action	a	`label (name a)`	`In` a
output action	$\overline{a}$	`label (coname a)`	`Out` a

Table 1: Syntax of CCS operators, constant and actions

The transition semantics of CCS processes follows Structural Operational Semantics (SOS) in Fig. 1:

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

The last 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. In its definition, `CCS_Subst` is a recursive substutiion function with the following long definition:

$\mathtt{CCS\_Subst}\ \mathtt{nil}\ E'\ X = \mathtt{nil}$
$\mathtt{CCS\_Subst}\ (u..E)\ E'\ X = u..\mathtt{CCS\_Subst}\ E\ E'\ X$
$\mathtt{CCS\_Subst}\ (E_1 + E_2)\ E'\ X =$
$\mathtt{CCS\_Subst}\ E_1\ E'\ X + \mathtt{CCS\_Subst}\ E_2\ E'\ X$
$\mathtt{CCS\_Subst}\ (E_1 \parallel E_2)\ E'\ X =$
$\mathtt{CCS\_Subst}\ E_1\ E'\ X \parallel \mathtt{CCS\_Subst}\ E_2\ E'\ X$
$\mathtt{CCS\_Subst}\ (\nu\ L\ E)\ E'\ X = \nu\ L\ (\mathtt{CCS\_Subst}\ E\ E'\ X)$

```
CCS_Subst (relab E f) E′ X = relab (CCS_Subst E E′ X) f
CCS_Subst (var Y) E′ X = if Y = X then E′ else var Y
CCS_Subst (rec Y E) E′ X =
if Y = X then rec Y E else rec Y (CCS_Subst E E′ X)                [CCS_Subst_def]
```

From HOL's viewpoint, these SOS rules are *inductive definitions* on the tenary relation `TRANS` of type "`(α, β) CCS →β Action →(α, β) CCS →bool`", generated by HOL's `Hol_reln` function.

A useful facility exploiting the interplay between HOL4 and Standard ML (that follows an idea by Nesi [8]) is a complex Standard ML function taking a CCS process and returning a theorem indicating all direct transitions of the process.[4] 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) there exists transitions from $(a.0 \mid \bar{a}.0)$ (optional); (2) the correctness for each of the transitions; and (3) 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: (`‘‘` is for double-backquotes of HOL terms, `>` is HOL's prompt)

```
> 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:

```
⊢ ∀u E.
      In “a”..nil ∥ Out “a”..nil −u→ E  ⟺
      ((u = In “a”) ∧ (E = nil ∥ Out “a”..nil) ∨
       (u = Out “a”) ∧ (E = In “a”..nil ∥ nil)) ∨
      (u = τ) ∧ (E = nil ∥ nil)                                  [Example.ex_A]
```

4.2. Context, guardedness and (pre)congruence

We have chosen to use λ-expressions (with the type "`(α, β) CCS →(α, β) CCS`") to represent *multi-hole contexts*. This choice has a significant advantage over *one-hole contexts*, as each hole corresponds to one appearance of the *same* variable in single-variable expressions (or equations). Thus *contexts* can be directly used in formulating the unique solution of equations theorems in single-variable cases. The precise definition is given inductively:

```
CONTEXT (λt. t)
CONTEXT (λt. p)
CONTEXT e ⇒ CONTEXT (λt. a..e t)
CONTEXT e₁ ∧ CONTEXT e₂ ⇒ CONTEXT (λt. e₁ t + e₂ t)
CONTEXT e₁ ∧ CONTEXT e₂ ⇒ CONTEXT (λt. e₁ t ∥ e₂ t)
CONTEXT e ⇒ CONTEXT (λt. ν L (e t))
CONTEXT e ⇒ CONTEXT (λt. relab (e t) rf)                           [CONTEXT_rules]
```

A context is *weakly guarded* (`WG`) if each hole is underneath a prefix:

```
WG (λt. p)
CONTEXT e ⇒ WG (λt. a..e t)
WG e₁ ∧ WG e₂ ⇒ WG (λt. e₁ t + e₂ t)
WG e₁ ∧ WG e₂ ⇒ WG (λt. e₁ t ∥ e₂ t)
WG e ⇒ WG (λt. ν L (e t))
WG e ⇒ WG (λt. relab (e t) rf)                                           [WG_rules]
```

[4]If the input process were infinite branching, due to the use of recursion or relabeling operators, the program will loop forever.

(Notice the differences between a weak guarded context and a normal one: $\lambda t.\, t$ is not weakly guarded as the variable is directly exposed without any prefixed action. And $\lambda t.\, a.\, e[t]$ is weakly guarded as long as $e[\cdot]$ is a context, not necessary weakly guarded.)

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

$\mathtt{SG}\ (\lambda\, t.\ p)$
$\mathtt{CONTEXT}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ \mathtt{label}\ l\,\mathtt{..}\,e\ t)$
$\mathtt{SG}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ a\,\mathtt{..}\,e\ t)$
$\mathtt{SG}\ e_1 \wedge \mathtt{SG}\ e_2 \Rightarrow \mathtt{SG}\ (\lambda\, t.\ e_1\ t + e_2\ t)$
$\mathtt{SG}\ e_1 \wedge \mathtt{SG}\ e_2 \Rightarrow \mathtt{SG}\ (\lambda\, t.\ e_1\ t \parallel e_2\ t)$
$\mathtt{SG}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ \nu\ L\ (e\ t))$
$\mathtt{SG}\ e \Rightarrow \mathtt{SG}\ (\lambda\, t.\ \mathtt{relab}\ (e\ t)\ rf)$ [SG_rules]

A context is *sequential* (SEQ) if each of its *subcontexts* with a hole, apart from the hole itself, is in forms of prefixes or sums: (c.f. Def. 3.3 and p.101,157 of [2] for the informal definitions.)

$\mathtt{SEQ}\ (\lambda\, t.\ t)$
$\mathtt{SEQ}\ (\lambda\, t.\ p)$
$\mathtt{SEQ}\ e \Rightarrow \mathtt{SEQ}\ (\lambda\, t.\ a\,\mathtt{..}\,e\ t)$
$\mathtt{SEQ}\ e_1 \wedge \mathtt{SEQ}\ e_2 \Rightarrow \mathtt{SEQ}\ (\lambda\, t.\ e_1\ t + e_2\ t)$ [SEQ_rules]

In the same manner, we can also define variants of contexts (GCONTEXT) and weakly guarded contexts (WGS) in which only guarded sums are allowed (i.e. arbitrary sums are forbidden):

$\mathtt{GCONTEXT}\ (\lambda\, t.\ t)$
$\mathtt{GCONTEXT}\ (\lambda\, t.\ p)$
$\mathtt{GCONTEXT}\ e \Rightarrow \mathtt{GCONTEXT}\ (\lambda\, t.\ a\,\mathtt{..}\,e\ t)$
$\mathtt{GCONTEXT}\ e_1 \wedge \mathtt{GCONTEXT}\ e_2 \Rightarrow \mathtt{GCONTEXT}\ (\lambda\, t.\ a_1\,\mathtt{..}\,e_1\ t + a_2\,\mathtt{..}\,e_2\ t)$
$\mathtt{GCONTEXT}\ e_1 \wedge \mathtt{GCONTEXT}\ e_2 \Rightarrow \mathtt{GCONTEXT}\ (\lambda\, t.\ e_1\ t \parallel e_2\ t)$
$\mathtt{GCONTEXT}\ e \Rightarrow \mathtt{GCONTEXT}\ (\lambda\, t.\ \nu\ L\ (e\ t))$
$\mathtt{GCONTEXT}\ e \Rightarrow \mathtt{GCONTEXT}\ (\lambda\, t.\ \mathtt{relab}\ (e\ t)\ rf)$ [GCONTEXT_rules]

$\mathtt{WGS}\ (\lambda\, t.\ p)$
$\mathtt{GCONTEXT}\ e \Rightarrow \mathtt{WGS}\ (\lambda\, t.\ a\,\mathtt{..}\,e\ t)$
$\mathtt{GCONTEXT}\ e_1 \wedge \mathtt{GCONTEXT}\ e_2 \Rightarrow \mathtt{WGS}\ (\lambda\, t.\ a_1\,\mathtt{..}\,e_1\ t + a_2\,\mathtt{..}\,e_2\ t)$
$\mathtt{WGS}\ e_1 \wedge \mathtt{WGS}\ e_2 \Rightarrow \mathtt{WGS}\ (\lambda\, t.\ e_1\ t \parallel e_2\ t)$
$\mathtt{WGS}\ e \Rightarrow \mathtt{WGS}\ (\lambda\, t.\ \nu\ L\ (e\ t))$
$\mathtt{WGS}\ e \Rightarrow \mathtt{WGS}\ (\lambda\, t.\ \mathtt{relab}\ (e\ t)\ rf)$ [WGS_rules]

Many lemmas have then to be proved for the 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.

A (pre)congruence is a relation on CCS processes defined on top of CONTEXT. The only difference between congruence and precongruence is that the former is an equivalence (reflexive, symmetric, transitive), while the latter can just be a preorder (reflexive, transitive):

$\mathtt{congruence}\ R \iff$
$\mathtt{equivalence}\ R\ \wedge$
$\forall\, x\ y\ ctx.\ \mathtt{CONTEXT}\ ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y)$ [congruence_def]

[precongruence_def]
$\mathtt{precongruence}\ R \iff$
$\mathtt{PreOrder}\ R \wedge \forall\, x\ y\ ctx.\ \mathtt{CONTEXT}\ ctx \Rightarrow R\ x\ y \Rightarrow R\ (ctx\ x)\ (ctx\ y)$

4.3. Bisimulation and Bisimilarity

An highlight in this project is the simplified definitions of bisimilarities using the new coinductive relational package for HOL. Without it, the bisimilaries can still be defined, but proving their basic properties would be more complicated.

The definition of strong bisimulation (`STRONG_BISIM`) follows its textbook definitions. Essentially it is a predicate (i.e. unary relation) stating what kind of binary relations on CCS process is a (strong) bisimulation:

$$
\begin{array}{l}
\texttt{STRONG\_BISIM}\ \mathit{Bsm} \iff \\
\forall E\ E'. \\
\quad \mathit{Bsm}\ E\ E' \Rightarrow \\
\quad \forall u. \\
\qquad (\forall E_1.\ E\ {-u\rightarrow}\ E_1 \Rightarrow \exists E_2.\ E'\ {-u\rightarrow}\ E_2 \land \mathit{Bsm}\ E_1\ E_2) \land \\
\qquad \forall E_2.\ E'\ {-u\rightarrow}\ E_2 \Rightarrow \exists E_1.\ E\ {-u\rightarrow}\ E_1 \land \mathit{Bsm}\ E_1\ E_2 \qquad [\texttt{STRONG\_BISIM}]
\end{array}
$$

With above definition it is easy to prove that the identity relation is indeed a bisimulation, that bisimulation is preserved by inversion, composition, and union operations.

Without the coinductive relation package for HOL, the definition would be the following one:

$$E \sim E' \iff \exists \mathit{Bsm}.\ \mathit{Bsm}\ E\ E' \land \texttt{STRONG\_BISIM}\ \mathit{Bsm} \qquad [\texttt{STRONG\_EQUIV}]$$

This is indeed the way followed by Nesi [8] at the time of HOL88. With the above definition, deriving, e.g., the property (*) below [2, p. 91] is tedious:

$P \sim Q$ iff, for all μ, (*)

(i) Whenever $P \xrightarrow{\mu} P'$ then, for some Q', $Q \xrightarrow{\mu} Q'$ and $P' \sim Q'$

(ii) Whenever $Q \xrightarrow{\mu} Q'$ then, for some P', $P \xrightarrow{\mu} P'$ and $P' \sim Q'$

With HOL's new coinductive relation package (`Hol_coreln` (since Kananaskis-11 release), it is possible to define (strong) bisimilarity in a more convienent way, which essentially amounts to defining bisimilarity as the greatest fixed-point of the appropriate functional on relations. Precisely we call `Hol_coreln` command in the following way:[5]

```
val (STRONG_EQUIV_rules, STRONG_EQUIV_coind, STRONG_EQUIV_cases) = Hol_coreln `
    (!(E :('a, 'b) CCS) (E' :('a, 'b) CCS).
       (!u.
         (!E1. TRANS E u E1 ==>
               (?E2. TRANS E' u E2 /\ STRONG_EQUIV E1 E2)) /\
         (!E2. TRANS E' u E2 ==>
               (?E1. TRANS E u E1 /\ STRONG_EQUIV E1 E2))) ==> STRONG_EQUIV E E')`;
```

`Hol_coreln` returns 3 theorems: the first one, `STRONG_EQUIV_rules`, is the same as the input term[6], but now promoted into a theorem. The second and third theorems, namely `STRONG_EQUIV_coind` and `STRONG_-EQUIV_cases`, represent the coinduction proof method for bisimilarity (i.e. any bisimulation is contained in bisimilarity) and the fixed-point property of bisimilarity (bisimilarity itself is a bisimulation, thus the largest bisimulation):

$$
\begin{array}{l}
1.\ \vdash (\forall u. \\
\qquad (\forall E_1.\ E\ {-u\rightarrow}\ E_1 \Rightarrow \exists E_2.\ E'\ {-u\rightarrow}\ E_2 \land E_1 \sim E_2) \land \\
\qquad \forall E_2.\ E'\ {-u\rightarrow}\ E_2 \Rightarrow \exists E_1.\ E\ {-u\rightarrow}\ E_1 \land E_1 \sim E_2) \Rightarrow \\
\quad E \sim E' \qquad [\texttt{STRONG\_EQUIV\_rules}]
\end{array}
$$

[5]Here `!` and `?` stand for universal and existential quantifiers in HOL's ASCII-based term syntax.

[6]Our mixing of HOL notation and mathematical notation in this paper is not arbitrary. We have pasted here the original proof scripts, written in HOL ASCII term notation (c.f. [23] for more details; HOL4 also supports writing Unicode symbols directly in proof scripts). All formal definitions and theorems in the paper are automatically generated from HOL4. We have tried to generate Unicode and TeX outputs as easy as possible to read (what is really arbitrary is the presense/absense of outermost universal quantifiers in all generated theorems).

2. $\vdash$ $(\forall a_0\ \ a_1\,.$
$\quad STRONG\_EQUIV'\ \ a_0\ \ a_1 \Rightarrow$
$\quad \forall u\,.$
$\quad\quad (\forall E_1\,.$
$\quad\quad\quad a_0\ -u\rightarrow\ E_1 \Rightarrow$
$\quad\quad\quad \exists E_2\,.\ a_1\ -u\rightarrow\ E_2 \wedge STRONG\_EQUIV'\ E_1\ E_2) \wedge$
$\quad\quad \forall E_2\,.$
$\quad\quad\quad a_1\ -u\rightarrow\ E_2 \Rightarrow$
$\quad\quad\quad \exists E_1\,.\ a_0\ -u\rightarrow\ E_1 \wedge STRONG\_EQUIV'\ E_1\ E_2) \Rightarrow$
$\forall a_0\ \ a_1\,.\ STRONG\_EQUIV'\ \ a_0\ \ a_1 \Rightarrow a_0 \sim a_1$ [STRONG_EQUIV_coind]

3. $\vdash$ $a_0 \sim a_1 \iff$
$\quad \forall u\,.$
$\quad\quad (\forall E_1\,.\ a_0\ -u\rightarrow\ E_1 \Rightarrow \exists E_2\,.\ a_1\ -u\rightarrow\ E_2 \wedge E_1 \sim E_2) \wedge$
$\quad\quad \forall E_2\,.\ a_1\ -u\rightarrow\ E_2 \Rightarrow \exists E_1\,.\ a_0\ -u\rightarrow\ E_1 \wedge E_1 \sim E_2$ [STRONG_EQUIV_cases]

The last theorem, `STRONG_EQUIV_cases`, is indeed property (*) mentioned earlier.

To define (weak) bisimilarity, we first need to define weak transitions of CCS processes. Following Nesi [8], we define a (possibly empty) sequence of τ-transitions between two processes as a new relation `EPS` ($\overset{\epsilon}{\Rightarrow}$), which is the reflexive transitive closure (RTC, denoted by * in HOL4) of ordinary τ-transitions of CCS processes:

`EPS` = $(\lambda E\ E'.\ E\ -\tau\rightarrow\ E')^*$ [EPS_def]

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

$E\ =u\Rightarrow\ E' \iff \exists E_1\ E_2\,.\ E \overset{\epsilon}{\Rightarrow} E_1 \wedge E_1\ -u\rightarrow\ E_2 \wedge E_2 \overset{\epsilon}{\Rightarrow} E'$ [WEAK_TRANS]

Weak bisimulation is defined upon weak transition:

`WEAK_BISIM` $Wbsm \iff$
$\forall E\ \ E'.$
$\quad Wbsm\ \ E\ \ E' \Rightarrow$
$\quad (\forall l\,.$
$\quad\quad (\forall E_1\,.$
$\quad\quad\quad E\ -\texttt{label}\ l\rightarrow\ E_1 \Rightarrow$
$\quad\quad\quad \exists E_2\,.\ E'\ =\texttt{label}\ l\Rightarrow\ E_2 \wedge Wbsm\ E_1\ E_2) \wedge$
$\quad\quad \forall E_2\,.$
$\quad\quad\quad E'\ -\texttt{label}\ l\rightarrow\ E_2 \Rightarrow \exists E_1\,.\ E\ =\texttt{label}\ l\Rightarrow\ E_1 \wedge Wbsm\ E_1\ E_2) \wedge$
$\quad (\forall E_1\,.\ E\ -\tau\rightarrow\ E_1 \Rightarrow \exists E_2\,.\ E' \overset{\epsilon}{\Rightarrow} E_2 \wedge Wbsm\ E_1\ E_2) \wedge$
$\quad \forall E_2\,.\ E'\ -\tau\rightarrow\ E_2 \Rightarrow \exists E_1\,.\ E \overset{\epsilon}{\Rightarrow} E_1 \wedge Wbsm\ E_1\ E_2$ [WEAK_BISIM]

Again, we can prove that the identity relation is a bisimulation, that bisimulation is preserved by inversion, composition, and union. The definition of weak bisimilarity can be generated with the following call to `Hol_coreln`:

```
val (WEAK_EQUIV_rules, WEAK_EQUIV_coind, WEAK_EQUIV_cases) = Hol_coreln ‘
    (!(E :('a, 'b) CCS) (E' :('a, 'b) CCS).
       (!l.
         (!E1. TRANS E  (label l) E1 ==>
               (?E2. WEAK_TRANS E' (label l) E2 /\ WEAK_EQUIV E1 E2)) /\
         (!E2. TRANS E' (label l) E2 ==>
               (?E1. WEAK_TRANS E  (label l) E1 /\ WEAK_EQUIV E1 E2))) /\
       (!E1. TRANS E  tau E1 ==> (?E2. EPS E' E2 /\ WEAK_EQUIV E1 E2)) /\
       (!E2. TRANS E' tau E2 ==> (?E1. EPS E  E1 /\ WEAK_EQUIV E1 E2))
      ==> WEAK_EQUIV E E')‘;
```

This returns the following 3 theorems defining `WEAK_EQUIV`:

1. $$\begin{array}{ll}
\vdash \forall E\ E'. & \\
\quad (\forall l. & \\
\qquad (\forall E_1. & \\
\qquad\quad E\ -\texttt{label}\ l\!\rightarrow\ E_1\ \Rightarrow & \\
\qquad\quad \exists E_2.\ E'\ =\texttt{label}\ l\!\Rightarrow\ E_2\ \wedge\ E_1\ \approx\ E_2)\ \wedge & \\
\qquad \forall E_2. & \\
\qquad\quad E'\ -\texttt{label}\ l\!\rightarrow\ E_2\ \Rightarrow\ \exists E_1.\ E\ =\texttt{label}\ l\!\Rightarrow\ E_1\ \wedge\ E_1\ \approx\ E_2)\ \wedge & \\
\quad (\forall E_1.\ E\ -\tau\!\rightarrow\ E_1\ \Rightarrow\ \exists E_2.\ E'\ \overset{\epsilon}{\Rightarrow}\ E_2\ \wedge\ E_1\ \approx\ E_2)\ \wedge & \\
\quad (\forall E_2.\ E'\ -\tau\!\rightarrow\ E_2\ \Rightarrow\ \exists E_1.\ E\ \overset{\epsilon}{\Rightarrow}\ E_1\ \wedge\ E_1\ \approx\ E_2)\ \Rightarrow & \\
\quad E\ \approx\ E' & \texttt{[WEAK\_EQUIV\_rules]}
\end{array}$$

2. $$\begin{array}{ll}
\vdash \forall\ \mathit{WEAK\_EQUIV}'. & \\
\quad (\forall a_0\ a_1. & \\
\qquad \mathit{WEAK\_EQUIV}'\ a_0\ a_1\ \Rightarrow & \\
\qquad (\forall l. & \\
\qquad\quad (\forall E_1. & \\
\qquad\qquad a_0\ -\texttt{label}\ l\!\rightarrow\ E_1\ \Rightarrow & \\
\qquad\qquad \exists E_2.\ a_1\ =\texttt{label}\ l\!\Rightarrow\ E_2\ \wedge\ \mathit{WEAK\_EQUIV}'\ E_1\ E_2)\ \wedge & \\
\qquad\quad \forall E_2. & \\
\qquad\qquad a_1\ -\texttt{label}\ l\!\rightarrow\ E_2\ \Rightarrow & \\
\qquad\qquad \exists E_1.\ a_0\ =\texttt{label}\ l\!\Rightarrow\ E_1\ \wedge\ \mathit{WEAK\_EQUIV}'\ E_1\ E_2)\ \wedge & \\
\qquad (\forall E_1.\ a_0\ -\tau\!\rightarrow\ E_1\ \Rightarrow\ \exists E_2.\ a_1\ \overset{\epsilon}{\Rightarrow}\ E_2\ \wedge\ \mathit{WEAK\_EQUIV}'\ E_1\ E_2)\ \wedge & \\
\qquad \forall E_2.\ a_1\ -\tau\!\rightarrow\ E_2\ \Rightarrow\ \exists E_1.\ a_0\ \overset{\epsilon}{\Rightarrow}\ E_1\ \wedge\ \mathit{WEAK\_EQUIV}'\ E_1\ E_2)\ \Rightarrow & \\
\quad \forall a_0\ a_1.\ \mathit{WEAK\_EQUIV}'\ a_0\ a_1\ \Rightarrow\ a_0\ \approx\ a_1 & \texttt{[WEAK\_EQUIV\_coind]}
\end{array}$$

3. $$\begin{array}{ll}
\vdash \forall a_0\ a_1. & \\
\quad a_0\ \approx\ a_1\ \iff & \\
\quad (\forall l. & \\
\qquad (\forall E_1. & \\
\qquad\quad a_0\ -\texttt{label}\ l\!\rightarrow\ E_1\ \Rightarrow & \\
\qquad\quad \exists E_2.\ a_1\ =\texttt{label}\ l\!\Rightarrow\ E_2\ \wedge\ E_1\ \approx\ E_2)\ \wedge & \\
\qquad \forall E_2. & \\
\qquad\quad a_1\ -\texttt{label}\ l\!\rightarrow\ E_2\ \Rightarrow & \\
\qquad\quad \exists E_1.\ a_0\ =\texttt{label}\ l\!\Rightarrow\ E_1\ \wedge\ E_1\ \approx\ E_2)\ \wedge & \\
\quad (\forall E_1.\ a_0\ -\tau\!\rightarrow\ E_1\ \Rightarrow\ \exists E_2.\ a_1\ \overset{\epsilon}{\Rightarrow}\ E_2\ \wedge\ E_1\ \approx\ E_2)\ \wedge & \\
\quad \forall E_2.\ a_1\ -\tau\!\rightarrow\ E_2\ \Rightarrow\ \exists E_1.\ a_0\ \overset{\epsilon}{\Rightarrow}\ E_1\ \wedge\ E_1\ \approx\ E_2 & \texttt{[WEAK\_EQUIV\_cases]}
\end{array}$$

The coinduction principle `WEAK_EQUIV_coind` says that any bisimulation is contained in the resulting relation (i.e. it is largest), but it didn't constrain the resulting relation in the set of fixed points (e.g. even the universal relation—the set of all pairs—would fit with this theorem); the purpose of `WEAK_EQUIV_cases` is to further assert that the resulting relation is indeed a fixed point. Thus `WEAK_EQUIV_coind` and `WEAK_EQUIV_cases` together make sure that bisimilarity is the greatest fixed point, as the former contributes to "greatest" while the latter contributes to "fixed point". Without HOL's coinductive relation package, (weak) bisimilarity would have to be defined by following literally Def. 2.1; then other properties of bisimilarity, such as the fixed-point property in `WEAK_EQUIV_cases`, would have to be derived manually.

Finally, the original definition of `WEAK_EQUIV` becomes a theorem:

$$\vdash E\ \approx\ E'\ \iff\ \exists\, \mathit{Wbsm}.\ \mathit{Wbsm}\ E\ E'\ \wedge\ \texttt{WEAK\_BISIM}\ \mathit{Wbsm} \qquad \texttt{[WEAK\_EQUIV]}$$

The formal definition of rooted bisimilarity ($\approx^{\mathrm{c}}$, `OBS_CONGR`) follows Definition 2.2:

$$\begin{array}{ll}
E\ \approx^{\mathrm{c}} E'\ \iff & \\
\forall u. & \\
\quad (\forall E_1.\ E\ -u\!\rightarrow\ E_1\ \Rightarrow\ \exists E_2.\ E'\ =u\!\Rightarrow\ E_2\ \wedge\ E_1\ \approx\ E_2)\ \wedge & \\
\quad \forall E_2.\ E'\ -u\!\rightarrow\ E_2\ \Rightarrow\ \exists E_1.\ E\ =u\!\Rightarrow\ E_1\ \wedge\ E_1\ \approx\ E_2 & \texttt{[OBS\_CONGR]}
\end{array}$$

Below is the formal version of Lemma 2.3, which is needed in the proof of unique-solution Theorem 3.14:

$$\begin{array}{l}
\vdash \texttt{WEAK\_BISIM}\ \mathit{Wbsm} \Rightarrow \\
\quad \forall E\ E'.\\
\qquad (\forall u.\\
\qquad\quad (\forall E_1.\ E\ {-u\rightarrow}\ E_1 \Rightarrow \exists E_2.\ E'\ {=u\Rightarrow}\ E_2 \land \mathit{Wbsm}\ E_1\ E_2) \land \\
\qquad\quad \forall E_2.\ E'\ {-u\rightarrow}\ E_2 \Rightarrow \exists E_1.\ E\ {=u\Rightarrow}\ E_1 \land \mathit{Wbsm}\ E_1\ E_2) \Rightarrow \\
\qquad E \approx^{\mathrm{c}} E' \qquad\qquad [\texttt{OBS\_CONGR\_BY\_WEAK\_BISIM}]
\end{array}$$

Both strong bisimilarity ($\sim$) and rooted bisimilarity ($\approx^c$) are congruence relations:

$\vdash$ `congruence STRONG_EQUIV` [STRONG_EQUIV_congruence]

$\vdash$ `congruence OBS_CONGR` [OBS_CONGR_congruence]

Although weak bisimilarity ($\approx$) is *not* a congruence with respect to `CONTEXT`, it is indeed substitutive with respect to `GCONTEXT` as $\approx$ is indeed preserved by weakly-guarded sums.

On the relationship between bisimilarity and rooted bisimilarity, we also prove Deng Lemma and Hennessy Lemma (Lemma 4.1 and 4.2 in [10]):

$$\begin{array}{l}
\vdash p \approx q \Rightarrow \\
\quad (\exists p'.\ p\ {-\tau\rightarrow}\ p' \land p' \approx q) \lor (\exists q'.\ q\ {-\tau\rightarrow}\ q' \land p \approx q') \lor \\
\quad p \approx^{\mathrm{c}} q \qquad\qquad [\texttt{DENG\_LEMMA}]
\end{array}$$

$$\vdash p \approx q \iff p \approx^{\mathrm{c}} q \lor p \approx^{\mathrm{c}} \tau..q \lor \tau..p \approx^{\mathrm{c}} q \qquad [\texttt{HENNESSY\_LEMMA}]$$

4.4. Algebraic Laws

Having formalised the definitions of strong bisimulation and strong bisimilarity, we can derive *algebraic laws* for the bisimilarities. We only report a few laws for the sum operator:

`STRONG_SUM_IDEMP:`	$\vdash E + E \sim E$
`STRONG_SUM_COMM:`	$\vdash E + E' \sim E' + E$
`STRONG_SUM_IDENT_L:`	$\vdash \texttt{nil} + E \sim E$
`STRONG_SUM_IDENT_R:`	$\vdash E + \texttt{nil} \sim E$
`STRONG_SUM_ASSOC_R:`	$\vdash E + E' + E'' \sim E + (E' + E'')$
`STRONG_SUM_ASSOC_L:`	$\vdash E + (E' + E'') \sim E + E' + E''$
`STRONG_SUM_MID_IDEMP:`	$\vdash E + E' + E \sim E' + E$
`STRONG_LEFT_SUM_MID_IDEMP:`	$\vdash E + E' + E'' + E' \sim E + E'' + E'$

The first 5 laws are proved by constructing bisimulation relations, and their formal proofs are written in a goal-directed manner. Instead, the last 3 laws are derived in a forward manner by applications of previous proven laws (without directly using the SOS inference rules and the definition of bisimulation).

The algebraic laws can be lifted to weak bisimilarity and rooted bisimilarity, as they are coarser than strong bisimilarity.

4.5. Expansion, Contraction and Rooted Contraction

To formally define bisimulation expansion and contraction (and their preorders), we have followed the same ways as in the case of strong and weak bisimilarities:

$$\begin{array}{l}
\texttt{EXPANSION}\ \mathit{Exp} \iff \\
\forall E\ E'.\\
\quad \mathit{Exp}\ E\ E' \Rightarrow \\
\quad (\forall l.\\
\qquad (\forall E_1.\\
\qquad\quad E\ {-\texttt{label}\ l\rightarrow}\ E_1 \Rightarrow \exists E_2.\ E'\ {-\texttt{label}\ l\rightarrow}\ E_2 \land \mathit{Exp}\ E_1\ E_2) \land
\end{array}$$

$\qquad\qquad\forall E_2.\ E' \ \texttt{-label}\ l\!\rightarrow\ E_2 \Rightarrow \exists E_1.\ E \ \texttt{=label}\ l\!\Rightarrow\ E_1 \wedge \mathit{Exp}\ E_1\ E_2) \wedge$
$\quad(\forall E_1.\ E\ {-\tau\!\rightarrow}\ E_1 \Rightarrow \mathit{Exp}\ E_1\ E' \vee \exists E_2.\ E'\ {-\tau\!\rightarrow}\ E_2 \wedge \mathit{Exp}\ E_1\ E_2) \wedge$
$\quad\forall E_2.\ E'\ {-\tau\!\rightarrow}\ E_2 \Rightarrow \exists E_1.\ E\ {=\tau\!\Rightarrow}\ E_1 \wedge \mathit{Exp}\ E_1\ E_2$ `[EXPANSION]`

$\vdash P \succeq_{\mathrm{e}} Q \iff \exists \mathit{Exp}.\ \mathit{Exp}\ P\ Q \wedge \texttt{EXPANSION}\ \mathit{Exp}$ `[expands_thm]`

$\texttt{CONTRACTION}\ \mathit{Con} \iff$
$\forall E\ E'.$
$\quad\mathit{Con}\ E\ E' \Rightarrow$
$\quad(\forall l.$
$\qquad(\forall E_1.$
$\qquad\qquad E\ \texttt{-label}\ l\!\rightarrow\ E_1 \Rightarrow \exists E_2.\ E'\ \texttt{-label}\ l\!\rightarrow\ E_2 \wedge \mathit{Con}\ E_1\ E_2) \wedge$
$\qquad\forall E_2.\ E'\ \texttt{-label}\ l\!\rightarrow\ E_2 \Rightarrow \exists E_1.\ E\ \texttt{=label}\ l\!\Rightarrow\ E_1 \wedge E_1 \approx E_2) \wedge$
$\quad(\forall E_1.\ E\ {-\tau\!\rightarrow}\ E_1 \Rightarrow \mathit{Con}\ E_1\ E' \vee \exists E_2.\ E'\ {-\tau\!\rightarrow}\ E_2 \wedge \mathit{Con}\ E_1\ E_2) \wedge$
$\quad\forall E_2.\ E'\ {-\tau\!\rightarrow}\ E_2 \Rightarrow \exists E_1.\ E \overset{\epsilon}{\Rightarrow} E_1 \wedge E_1 \approx E_2$ `[CONTRACTION]`

$\vdash P \succeq_{\mathrm{bis}} Q \iff \exists \mathit{Con}.\ \mathit{Con}\ P\ Q \wedge \texttt{CONTRACTION}\ \mathit{Con}$ `[contracts_thm]`

We can prove that the contraction preorder is contained in weak bisimilarity, and contains the expansion preorder.

Proposition 4.1. *(Relationships between contraction, expansion and weak bisimilarity)*

1. *('expansion' implies 'contraction')*
   $\vdash P \succeq_{\mathrm{e}} Q \Rightarrow P \succeq_{\mathrm{bis}} Q$ `[expands_IMP_contracts]`
2. *('contraction' implies weak bisimilarity)*
   $\vdash P \succeq_{\mathrm{bis}} Q \Rightarrow P \approx Q$ `[contracts_IMP_WEAK_EQUIV]`

In general a proof of a property for the contraction preorder is harder than that for the expansion preorder; we sometimes need to reduce the proof to corresponding properties of weak bisimilarity. Also, surprisingly, a contraction does not imply weak bisimulation, i.e. the following proposition doesn't hold:

$\forall \mathit{Con}.\ \texttt{CONTRACTION}\ \mathit{Con} \Rightarrow \texttt{WEAK\_BISIM}\ \mathit{Con}$

As a result, to finish the proof of `contracts_IMP_WEAK_EQUIV`, we do not prove that *Con* itself is a weak bisimulation, but rather that the union of *Con* and weak bisimilarity is a weak bisimulation.

The rooted contraction ($\succeq^{\mathrm{c}}_{\mathrm{bis}}$, `OBS_contracts`) is formally defined as follows, with its precongruence result:

$E \succeq^{\mathrm{c}}_{\mathrm{bis}} E' \iff$
$\forall u.$
$\quad(\forall E_1.\ E\ {-u\!\rightarrow}\ E_1 \Rightarrow \exists E_2.\ E'\ {-u\!\rightarrow}\ E_2 \wedge E_1 \succeq_{\mathrm{bis}} E_2) \wedge$
$\quad\forall E_2.\ E'\ {-u\!\rightarrow}\ E_2 \Rightarrow \exists E_1.\ E\ {=u\!\Rightarrow}\ E_1 \wedge E_1 \approx E_2$ `[OBS_contracts]`

$\vdash$ `precongruence OBS_contracts` `[OBS_contracts_precongruence]`

4.6. The formalisation of "bisimulation up to"

"Bisimulation up to" is a family of powerful proof techniques, used to reduce the size of a relation needed to define a bisimulation. By definition, two processes are bisimilar if there exists a bisimulation relation containing them as a pair. However, in practice this definition is hardly ever followed plainly; instead, to reduce the size of the relations exhibited one prefers to define relations which are bisimulations only when closed up under some specific and privileged relation, so to relieve the proof work needed. These are called an *"up-to" techniques.*

We recall that we often write $P\mathcal{R}Q$ to mean $(P, Q) \in \mathcal{R}$, for any binary relation $\mathcal{R}$. Moreover, $\sim \mathcal{S} \sim$ is the composition of the binary relations, hence $P \sim \mathcal{S} \sim Q$ means that, for some P' and Q', we have $P \sim P'$, $P'\,\mathcal{S}\,Q'$ and $Q' \sim Q$.

Definition 4.2 (Bisimulation up to $\sim$). *$\mathcal{S}$ is a "*bisimulation up to $\sim$*" if $P\mathcal{S}Q$ implies, for all μ,*

1. *Whenever $P \overset{\mu}{\to} P'$ then, for some Q', $Q \overset{\mu}{\to} Q'$ and $P' \sim \mathcal{S} \sim Q'$,*
2. *Whenever $Q \overset{\mu}{\to} Q'$ then, for some P', $P \overset{\mu}{\to} P'$ and $P' \sim \mathcal{S} \sim Q'$.*

Formally,

```
STRONG_BISIM_UPTO $Bsm$ $\iff$
$\forall E\ E'$.
    $Bsm\ E\ E'$ $\Rightarrow$
    $\forall u$.
        ($\forall E_1$.
            $E$ $-u\rightarrow$ $E_1$ $\Rightarrow$
            $\exists E_2$.
                $E'$ $-u\rightarrow$ $E_2$ $\wedge$
                (STRONG_EQUIV $\circ_r$ $Bsm$ $\circ_r$ STRONG_EQUIV) $E_1$ $E_2$) $\wedge$
        $\forall E_2$.
            $E'$ $-u\rightarrow$ $E_2$ $\Rightarrow$
            $\exists E_1$.
                $E$ $-u\rightarrow$ $E_1$ $\wedge$
                (STRONG_EQUIV $\circ_r$ $Bsm$ $\circ_r$ STRONG_EQUIV) $E_1$ $E_2$          [STRONG_BISIM_UPTO]
```

Theorem 4.3. *If $\mathcal{S}$ is a "bisimulation up to $\sim$", then $\mathcal{S} \subseteq \sim$:*

```
⊢ STRONG_BISIM_UPTO $Bsm$ $\wedge$ $Bsm\ P\ Q$ $\Rightarrow$ $P \sim Q$          [STRONG_EQUIV_BY_BISIM_UPTO]
```

Hence, to prove $P \sim Q$, one only needs to find a bisimulation up to $\sim$ that contains (P, Q). For weak bisimilarity, there are a few versions of the 'up-to bisimilarity'. The most common one is the following.

Definition 4.4. *(Bisimulation up to $\approx$) $\mathcal{S}$ is a "*bisimulation up to $\approx$*" if $P\mathcal{S}Q$ implies, for all μ,*

1. *Whenever $P \overset{\mu}{\to} P'$ then, for some Q', $Q \overset{\hat{\mu}}{\Rightarrow} Q'$ and $P' \sim \mathcal{S} \approx Q'$,*
2. *Whenever $Q \overset{\mu}{\to} Q'$ then, for some P', $P \overset{\hat{\mu}}{\Rightarrow} P'$ and $P' \approx \mathcal{S} \sim Q'$.*

Formally,

```
WEAK_BISIM_UPTO $Wbsm$ $\iff$
$\forall E\ E'$.
    $Wbsm\ E\ E'$ $\Rightarrow$
    ($\forall l$.
        ($\forall E_1$.
            $E$ $-$label $l\rightarrow$ $E_1$ $\Rightarrow$
            $\exists E_2$.
                $E'$ $=$label $l\Rightarrow$ $E_2$ $\wedge$
                (WEAK_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ STRONG_EQUIV) $E_1$ $E_2$) $\wedge$
        $\forall E_2$.
            $E'$ $-$label $l\rightarrow$ $E_2$ $\Rightarrow$
            $\exists E_1$.
                $E$ $=$label $l\Rightarrow$ $E_1$ $\wedge$
                (STRONG_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ WEAK_EQUIV) $E_1$ $E_2$) $\wedge$
    ($\forall E_1$.
        $E$ $-\tau\rightarrow$ $E_1$ $\Rightarrow$
        $\exists E_2$.
            $E'$ $\overset{\epsilon}{\Rightarrow}$ $E_2$ $\wedge$ (WEAK_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ STRONG_EQUIV) $E_1$ $E_2$) $\wedge$
    $\forall E_2$.
        $E'$ $-\tau\rightarrow$ $E_2$ $\Rightarrow$
        $\exists E_1$. $E$ $\overset{\epsilon}{\Rightarrow}$ $E_1$ $\wedge$ (STRONG_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ WEAK_EQUIV) $E_1$ $E_2$      [WEAK_BISIM_UPTO]
```

Theorem 4.5. *If $\mathcal{S}$ is a bisimulation up to $\approx$, then $\mathcal{S} \subseteq \approx$:*

$\vdash$ `WEAK_BISIM_UPTO` $Bsm \land Bsm\ P\ Q \Rightarrow P \approx Q$ `[WEAK_EQUIV_BY_BISIM_UPTO]`

The above version of "bisimulation up to $\approx$" is not powerful enough to be used within the proof of Milner's "unique solution of equations" theorem for $\approx$, for which we need the following version (though less practical to use, as checking multi-step transitions is usually harder then checking one-step transitions):

```
WEAK_BISIM_UPTO_ALT $Wbsm$ $\iff$
$\forall E\ E'$.
    $Wbsm\ E\ E' \Rightarrow$
    ($\forall l$.
         ($\forall E_1$.
              $E$ =label $l$⇒ $E_1 \Rightarrow$
              $\exists E_2$.
                  $E'$ =label $l$⇒ $E_2 \land$
                  (WEAK_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ WEAK_EQUIV) $E_1\ E_2$) $\land$
         $\forall E_2$.
             $E'$ =label $l$⇒ $E_2 \Rightarrow$
             $\exists E_1$.
                 $E$ =label $l$⇒ $E_1 \land$
                 (WEAK_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ WEAK_EQUIV) $E_1\ E_2$) $\land$
    ($\forall E_1$.
         $E$ =$\tau$⇒ $E_1 \Rightarrow$
         $\exists E_2$. $E' \overset{\epsilon}{\Rightarrow} E_2 \land$ (WEAK_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ WEAK_EQUIV) $E_1\ E_2$) $\land$
    $\forall E_2$.
        $E'$ =$\tau$⇒ $E_2 \Rightarrow$
        $\exists E_1$. $E \overset{\epsilon}{\Rightarrow} E_1 \land$ (WEAK_EQUIV $\circ_r$ $Wbsm$ $\circ_r$ WEAK_EQUIV) $E_1\ E_2$    [WEAK_BISIM_UPTO_ALT]
```

$\vdash$ `WEAK_BISIM_UPTO_ALT` $Bsm \land Bsm\ P\ Q \Rightarrow P \approx Q$ `[WEAK_EQUIV_BY_BISIM_UPTO_ALT]`

Finally, we present a fourth version, "bisimulation up to $\approx^{c}$" (`OBS_BISIM_UPTO`), which can be used as a proof method for rooted bisimilarity. (Again, it is not powerful enough to be used within the proof of Milner's "unique solution of equations" theorem for $\approx^{c}$.)

```
OBS_BISIM_UPTO $Obsm$ $\iff$
$\forall E\ E'$.
    $Obsm\ E\ E' \Rightarrow$
    $\forall u$.
        ($\forall E_1$.
             $E$ −$u$→ $E_1 \Rightarrow$
             $\exists E_2$.
                 $E'$ =$u$⇒ $E_2 \land$
                 (WEAK_EQUIV $\circ_r$ $Obsm$ $\circ_r$ STRONG_EQUIV) $E_1\ E_2$) $\land$
        $\forall E_2$.
            $E'$ −$u$→ $E_2 \Rightarrow$
            $\exists E_1$.
                $E$ =$u$⇒ $E_1 \land$
                (STRONG_EQUIV $\circ_r$ $Obsm$ $\circ_r$ WEAK_EQUIV) $E_1\ E_2$          [OBS_BISIM_UPTO]
```

$\vdash$ `OBS_BISIM_UPTO` $Obsm \land Obsm\ P\ Q \Rightarrow P \approx^{c} Q$ `[OBS_CONGR_BY_BISIM_UPTO]`

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

In this subsection we discuss two proofs of the fundamental result stating that rooted bisimilarity is the coarsest congruence contained in bisimilarity [11, 10, 2]:

$$\forall p\ \ q.\ p \approx^{\mathrm{c}} q \iff (\forall r.\ p + r \approx q + r)\ . \tag{3}$$

The coarsest congruence contained in (weak) bisimilarity, namely *bisimilarity congruence*, is the *composition closure* (`CC`) of (weak) bisimilarity:

`WEAK_CONGR = CC WEAK_EQUIV` [WEAK_CONGR]
`CC` R = ($\lambda g\ h.\ \forall c.$ `CONTEXT` $c \Rightarrow R\ (c\ g)\ (c\ h)$) [CC_def]

(We need not put $R\ g\ h$ in the antecedent of `CC_def`, as this is anyhow obtained from the trivial context $(\lambda x.\ x)$.) For any relation R on CCS processes, the composition closure of R is always at least as fine as R (here $\subseteq_r$ stands for *relational subset*)

$\vdash \forall R.$ `CC` $R \subseteq_r R$ [CC_is_finer]

Furthermore, we prove that any (pre)congruence contained in R (which itself may not be) is contained in the composition closure of R (hence the closure is the coarsest one):

$\vdash \forall R\ R'.$ `congruence` $R' \wedge R' \subseteq_r R \Rightarrow R' \subseteq_r$ `CC` R [CC_is_coarsest]
$\vdash \forall R\ R'.$ `precongruence` $R' \wedge R' \subseteq_r R \Rightarrow R' \subseteq_r$ `CC` R [CC_is_coarsest']

Given the central role of the sum operator, we also consider the closure of bisimilarity under such operator, called *equivalence compatible with sums* (`SUM_EQUIV`):

`SUM_EQUIV` = ($\lambda p\ q.\ \forall r.\ p + r \approx q + r$) [SUM_EQUIV]

Rooted bisimilarity $\approx^{\mathrm{c}}$ (a congruence contained in $\approx$), is now contained in `WEAK_CONGR`, which in turn is trivially contained in `SUM_EQUIV`, as shown in Fig. 2. Thus, to prove (3), the crux is to prove that `SUM_EQUIV` implies rooted bisimilarity ($\approx^{\mathrm{c}}$), making all three relations ($\approx^{\mathrm{c}}$, `WEAK_CONGR` and `SUM_EQUIV`) coincide:

$$\forall p\ \ q.\ (\forall r.\ p + r \approx q + r) \Rightarrow p \approx^{\mathrm{c}} q\ . \tag{4}$$

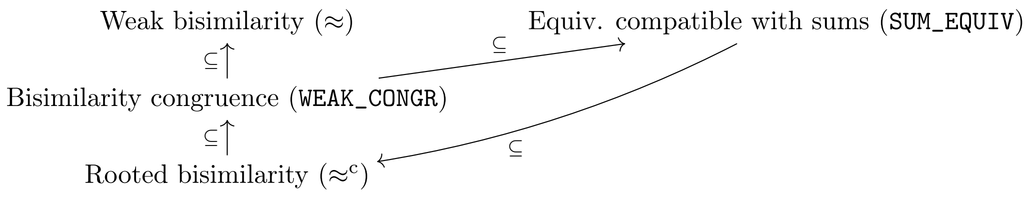
$$\begin{tikzcd}
\text{Weak bisimilarity } (\approx) & \text{Equiv. compatible with sums } (\texttt{SUM\_EQUIV}) \arrow[ddl, bend left, "\subseteq"] \\
\text{Bisimilarity congruence } (\texttt{WEAK\_CONGR}) \arrow[u, "\subseteq"] \arrow[ur, "\subseteq"] & \\
\text{Rooted bisimilarity } (\approx^{\mathrm{c}}) \arrow[u, "\subseteq"] &
\end{tikzcd}$$

Figure 2: Relationship between the equivalences mentioned

The standard argument [2] requires that p and q do not use up available labels (i.e. visible actions). Formalising such an argument requires a detailed treatment on free and bound names of CCS processes (with the restriction operator being a binder). However, the proof of (4) can be carried out just assuming that all immediate *weak* derivatives of p and q do not use all available labels. We have formalised this property and called it the *free action* property:

`free_action` $p \iff \exists a.\ \forall p'.\ \neg(p$ `=label` $a \Rightarrow p')$ [free_action_def]

With this property, the actual formalisation of (4) says:

$\vdash$ `free_action` $p\ \wedge$ `free_action` $q \Rightarrow (\forall r.\ p + r \approx q + r) \Rightarrow p \approx^{\mathrm{c}} q$ [COARSEST_CONGR_RL]

With an almost identical proof, rooted contraction ($\succeq^{\mathrm{c}}_{\mathrm{bis}}$) is also the coarsest precongruence contained in the bisimilarity contraction ($\succeq_{\mathrm{bis}}$) (the other direction is trivial):

[COARSEST_PRECONGR_RL]

$\vdash$ free_action $p \land$ free_action $q \Rightarrow (\forall r.\ p + r \succeq_{\mathrm{bis}} q + r) \Rightarrow p \succeq^{\mathrm{c}}_{\mathrm{bis}} q$

The formal proofs of the above two results follow Milner's proof [2]. The assumption is Milner's result requires $\mathrm{fn}(p) \cup \mathrm{fn}(q) \neq \mathscr{L}$ (where fn stands for *free names*), and is therefore strictly stronger than the assumption in our result above, where we only look at immediate weak derivatives p and q, requiring that there is at least one action that never appears as a label; moreover such an action need not be the same for p and for q (and where an action a and its complementary $\overline{a}$ are considered *different* actions).

There exists a different, more complex, proof of (3), given by van Glabbeek [11], which does not require any additional assumption. The core lemma says, for any two processes p and q, if there exists a *stable* (i.e. τ-free) process k which is not bisimilar with any derivative of p and q, then SUM_EQUIV indeed implies rooted bisimilarity ($\approx^{\mathrm{c}}$):

$\vdash (\exists k.$
$\quad$ STABLE $k \land (\forall p'\ u.\ p =u\Rightarrow p' \Rightarrow \neg(p' \approx k)) \land$
$\quad \forall q'\ u.\ q =u\Rightarrow q' \Rightarrow \neg(q' \approx k)) \Rightarrow$
$(\forall r.\ p + r \approx q + r) \Rightarrow$
$p \approx^{\mathrm{c}} q$ [PROP3_COMMON]

STABLE $p \iff \forall u\ p'.\ p\ -u\rightarrow\ p' \Rightarrow u \neq \tau$ [STABLE]

To construct this process k, the proof relies on arbitrary infinite sums of processes and uses transfinite induction to obtain an arbitrary large sequence of processes (firstly introduced by Jan Willem Klop [11]) that are all pairwise non-bisimilar. We have partially formalised this proof, because the typed logic implemented in various HOL systems (including Isabelle/HOL) is not strong enough to define a type for all possible ordinals [24], thus 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). The proof, in the section below, uses extensively HOL's pred_set theory [25], with a mix of CCS and pure mathematics.

4.8. Arbitrary many non-bisimilar processes

The assumption in the previous PROP3_COMMON requires the existence of a special CCS process, which is not weakly bisimilar to any sub-process emanating from the two root processes by weak transitions. In the worst case, there may be infinite such sub-processes (even on finitely branching processes). We can however consider the equivalence classes of CCS processes modulo weak bisimilarity. If there are infinitely many of such classes, then it will be possible to choose one that is distinct from all the (finitely many) states in the graphs of the two given processes. This can be done following Klop's contruction [11]. We call "Klop processes" the processes in this construction:

Definition 4.6 (Klop processes)**.** *For each ordinal λ, and an arbitrary chosen non-τ action a, define a CCS process k_λ as follows:*

- $k_0 = 0$,
- $k_{\lambda+1} = k_\lambda + a.\,k_\lambda$ *and*
- *for λ a limit ordinal, $k_\lambda = \sum_{\mu<\lambda} k_\mu$, meaning that k_λ is constructed from all graphs k_μ for $\mu < \lambda$ by identifying their root.*

When processes are finite-state, that is, the number of states in which a process may evolve by performing transitions is finite, we can use the following subset of Klop processes, defined as a recursive function (on natural numbers) in HOL4:

Definition 4.7. *(Klop processes as recursive function on natural numbers)*

KLOP a 0 = nil
KLOP a (SUC n) = KLOP a n + label a..KLOP a n [KLOP_def]

By induction on the definition of Klop processes, and using the SOS inference rules (Sum_1) and (Sum_2), we can prove the following properties of Klop functions:

Proposition 4.8. *(Properties of Klop functions and processes)*

1. *(All Klop processes are stable)*

   $\vdash$ STABLE (KLOP a n) [KLOP_PROP0]

2. *(Any transition from a Klop process leads to a smaller Klop process, and conversely)*

   $\vdash$ KLOP a n $-$label $a\rightarrow$ E $\iff$ $\exists m.\ m < n \land$ (E = KLOP a m) [KLOP_PROP1]

3. *(The weak version of the previous property)*

   $\vdash$ KLOP a n =label $a\Rightarrow$ E $\iff$ $\exists m.\ m < n \land$ (E = KLOP a m) [KLOP_PROP1']

4. *(All Klop processes are distinct according to strong bisimilarity)*

   $\vdash$ $m < n \Rightarrow \neg$(KLOP a m $\sim$ KLOP a n) [KLOP_PROP2]

5. *(All Klop processes are distinct according to weak bisimilarity)*

   $\vdash$ $m < n \Rightarrow \neg$(KLOP a m $\approx$ KLOP a n) [KLOP_PROP2']

6. *(Klop functions are one-one)*

   $\vdash$ ONE_ONE (KLOP a) [KLOP_ONE_ONE]

Having a recursive function on all natural numbers, we can map these into a set containing all Klop processes. We can therefore choose a number that is mapped onto a Klop process that is distinct (i.e., not bisimilar) from any of the derivatives of two given two finite-state CCS processes p and q (by definition, the number of such derivatives is finite). This is done by appealing to a basic set-theory result.

Lemma 4.9. *Given an equivalence relation R defined on a type, and two sets A, B of elements in this type, if A is finite, B is infinite, and all elements in B belong to distinct equivalence classes, then there exists an element k in B which is not equivalent to any element in A:*

$\vdash$ equivalence R $\Rightarrow$
  FINITE A $\land$ INFINITE B $\land$
  $(\forall x\ y.\ x \in B \land y \in B \land x \neq y \Rightarrow \neg R\ x\ y) \Rightarrow$
  $\exists k.\ k \in B \land \forall n.\ n \in A \Rightarrow \neg R\ n\ k$ [INFINITE_EXISTS_LEMMA]

To reason about finite-state CCS, we also need to define the concept of "finite-state":

Definition 4.10. *(Definitions related to finite-state CCS)*

1. *Define* reachable *as the RTC of a relation, which indicates the existence of a transition between two processes:*

   Reach = $(\lambda E\ E'.\ \exists u.\ E\ -u\rightarrow\ E')^*$ [Reach_def]

2. *The "nodes" of a process is the set of all processes reachable from it:*

   NODES p = $\{q$ | Reach p $q\}$ [NODES_def]

3. *A process is finite-state if the set of nodes is finite:*

   finite_state p $\iff$ FINITE (NODES p) [finite_state_def]

Among many properties of above definitions, we mainly rely on the following (trivial) property for weak transitions:

Proposition 4.11. *If p has a weak transition onto q, then q must be in the node set of p:*

$\vdash\ p\ {=}u{\Rightarrow}\ q \Rightarrow q \in$ `NODES` p `[WEAK_TRANS_IN_NODES]`

Using all the above results, now we can prove the following finite version of "Klop lemma":

Lemma 4.12. *(Klop lemma, finite version) For any two finite-state CCS p and q, there is another process k, which is not weakly bisimilar with any weak derivative of p and q (i.e., any process reachable from p or q by means of transitions):*

$$\begin{array}{l}\vdash \forall p\ q.\\ \quad \texttt{finite\_state}\ p \land \texttt{finite\_state}\ q \Rightarrow\\ \quad \exists k.\\ \qquad \texttt{STABLE}\ k \land (\forall p'\ u.\ p\ {=}u{\Rightarrow}\ p' \Rightarrow \neg(p' \approx k)) \land\\ \qquad \forall q'\ u.\ q\ {=}u{\Rightarrow}\ q' \Rightarrow \neg(q' \approx k) \qquad \texttt{[KLOP\_LEMMA\_FINITE]}\end{array}$$

Combining the above lemma with `PROP3_COMMON` and `COARSEST_CONGR_RL`, we can prove the following theorem for finite-state CCS:

Theorem 4.13. *(Coarsest congruence contained in $\approx$ for finite-state CCS)*

$$\begin{array}{l}\vdash \texttt{finite\_state}\ p \land \texttt{finite\_state}\ q \Rightarrow\\ \quad (p \approx^{c} q \iff \forall r.\ p + r \approx q + r) \qquad \texttt{[COARSEST\_CONGR\_FINITE]}\end{array}$$

4.9. Unique solution of equations

Strong bisimularity. Based on results about "bisimulation up to $\sim$", we have first proved the following lemma. It states that if X is weakly guarded in E, then the "first move" of E is independent of the agent substituted for X:

Lemma 4.14 (Lemma 3.13 of [2])**.** *If the variable X are weakly guarded in E, and $E\{P/X\} \overset{\alpha}{\to} P'$, then P' takes the form $E'\{P/X\}$ (for some expression E'), and moreover, for any Q, $E\{Q/X\} \overset{\alpha}{\to} E'\{Q/X\}$:*

`[STRONG_UNIQUE_SOLUTION_LEMMA]`

$$\begin{array}{l}\vdash \texttt{WG}\ E \Rightarrow\\ \quad \forall P\ a\ P'.\\ \qquad E\ P\ {-}a{\rightarrow}\ P' \Rightarrow\\ \qquad \exists E'.\ \texttt{CONTEXT}\ E' \land (P' = E'\ P) \land \forall Q.\ E\ Q\ {-}a{\rightarrow}\ E'\ Q\end{array}$$

Theorem 4.15 (Proposition 3.14 of [2])**.** *Suppose the expression E contains at most the variable X, and let X be weakly guarded in E.*

$$\textit{If } P \sim E\{P/X\} \textit{ and } Q \sim E\{Q/X\} \textit{ then } P \sim Q. \tag{5}$$

$\vdash$ `WG` $E \Rightarrow \forall P\ Q.\ P \sim E\ P \land Q \sim E\ Q \Rightarrow P \sim Q$ `[STRONG_UNIQUE_SOLUTION]`

In above proof, we have identified 14 major sub-goals, dividing them into 7 groups, in which each pair is symmetric (thus having similar proof steps). The proof of this last theorem consists of 500 lines (each line usually has 2 or 3 HOL tactics, to make the proof shorter).

Weak bisimilarity. Milner's book [2] contains only two "unique solutions of equations" theorems, one for strong equivalence, the other for observational congruence. There is however a version of the result for weak bisimilarity that shares a large portion of proof steps with that case for observational congruence. As weak bisimilarity is not congruence, we have to be more restrictive on the syntax for the equation, using strong guardedness.

Lemma 4.16. *If the variable X is strongly guarded and sequential in G, and $G\{P/X\} \overset{\alpha}{\to} P'$, then P' takes the form $H\{P/X\}$, for some expression H, and for any Q, we have $G\{Q/X\} \overset{\alpha}{\to} H\{Q/X\}$. Moreover H is sequential, and if $\alpha = \tau$ then H is also guarded.*

```
⊢ SG G ∧ GSEQ G ⇒
  ∀P a P′.
      G P −a→ P′ ⇒
      ∃H.
          GSEQ H ∧ ((a = τ) ⇒ SG H) ∧ (P′ = H P) ∧
          ∀Q. G Q −a→ H Q                          [WEAK_UNIQUE_SOLUTION_LEMMA]
```

We can then derive the target theorem:

Theorem 4.17. *(Unique solution of equations for weak bisimilarity) Let E be guarded and sequential expressions, and let $P \approx E\{P/X\}$, $Q \approx E\{Q/X\}$. Then $P \approx Q$.*

```
⊢ SG E ∧ GSEQ E ⇒ ∀P Q. P ≈ E P ∧ Q ≈ E Q ⇒ P ≈ Q        [WEAK_UNIQUE_SOLUTION]
```

Rooted bisimilarity. For the "unique solutions of equations" theorem of rooted bisimilarity, we use a different technique, represented by the following lemma. It is assertion OBS_CONGR_BY_WEAK_BISIM mentioned in Section 4.3, and corresponds to Lemma 2.3; we only report the text for OBS_CONGR_BY_WEAK_BISIM.

Lemma 4.18.
```
⊢ WEAK_BISIM Wbsm ⇒
  ∀E E′.
      (∀u.
          (∀E₁. E −u→ E₁ ⇒ ∃E₂. E′ =u⇒ E₂ ∧ Wbsm E₁ E₂) ∧
          ∀E₂. E′ −u→ E₂ ⇒ ∃E₁. E =u⇒ E₁ ∧ Wbsm E₁ E₂) ⇒
      E ≈ᶜ E′                                          [OBS_CONGR_BY_WEAK_BISIM]
```

This lemma plays a key role in the corresponding 'unique solutions' theorem, in connection with the following result. In this part, we have not used "bisimulation up-to" techniques. Using the above OBS_-CONGR_BY_WEAK_BISIM, we directly construct the appropriate bisimulation relations.

Lemma 4.19. *If the variable X is strongly guarded and sequential in G, and $G\{P/X\} \overset{\alpha}{\to} P'$, then P' takes the form $H\{P/X\}$, for some H, and for any Q, we also have $G\{Q/X\} \overset{\alpha}{\to} H\{Q/X\}$. Moreover H is sequential, and if $\alpha = \tau$ then H is also guarded.*

```
⊢ SG G ∧ SEQ G ⇒
  ∀P a P′.
      G P −a→ P′ ⇒
      ∃H.
          SEQ H ∧ ((a = τ) ⇒ SG H) ∧ (P′ = H P) ∧
          ∀Q. G Q −a→ H Q                           [OBS_UNIQUE_SOLUTION_LEMMA]
```

Theorem 4.20. *(Unique solution of equations for observational congruence) Let E be guarded and sequential expressions, and let $P \approx^c E\{P/X\}$, $Q \approx^c E\{Q/X\}$. Then $P \approx^c Q$.*

```
⊢ SG E ∧ SEQ E ⇒ ∀P Q. P ≈ᶜ E P ∧ Q ≈ᶜ E Q ⇒ P ≈ᶜ Q        [OBS_UNIQUE_SOLUTION]
```

4.10. Unique solution of contractions

A delicate point in the formalisation of the results about unique solution of contractions are the proof of Lemma 3.13 and lemmas alike; 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 concept of label-accumulated reflexive transitive closure (`LRTC`). Given a labeled transition relation `R` on CCS, `LRTC R` is a label-accumulated relation representing the trace of transitions:

```
LRTC R a l b ⟺
∀P.
    (∀x. P x [] x) ∧
    (∀x h y t z. R x h y ∧ P y t z ⇒ P x (h::t) z) ⇒
    P a l b                                                    [LRTC_DEF]
```

The trace relation for CCS can be then obtained by calling `LRTC` on the (strong, or single-step) labeled transition relation `TRANS` ($\xrightarrow{\mu}$) defined by SOS rules:

```
TRACE = LRTC TRANS                                             [TRACE_def]
```

If the list of actions is empty, that means that there is no transition and hence, if there is at most one visible action (i.e., a label) in the list of actions, then the trace is also a weak transition. Here we have to distinguish between two cases: no label and unique label (in the list of actions). The definition of "no label" in an action list is easy (here `MEM` tests if a given element is a member of a list):

```
NO_LABEL L ⟺ ¬∃l. MEM (label l) L                              [NO_LABEL_def]
```

The definition of "unique label" can be done in many ways, the following definition (a suggestion from Robert Beers) avoids any counting or filtering in the list. It says that a label is unique in a list of actions if and only if there is no label in the rest of list:

```
UNIQUE_LABEL u L ⟺
∃L₁ L₂. (L₁ ++ [u] ++ L₂ = L) ∧ NO_LABEL L₁ ∧ NO_LABEL L₂       [UNIQUE_LABEL_def]
```

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, a weak transition $P \overset{u}{\Rightarrow} P'$ is also a trace $P \xrightarrow{acts} P'$ with a non-empty action list *acts*, in which either there is no label (for $u = \tau$), or u is the unique label (for $u \neq \tau$):

```
⊢ P =u⇒ P' ⟺
  ∃acts.
      TRACE P acts P' ∧ ¬NULL acts ∧
      if u = τ then NO_LABEL acts else UNIQUE_LABEL u acts     [WEAK_TRANS_AND_TRACE]
```

Now the formalised version of Lemma 3.9: `[UNIQUE_SOLUTION_OF_CONTRACTIONS_LEMMA]`

```
⊢ (∃E. WGS E ∧ P ⪰bis E P ∧ Q ⪰bis E Q) ⇒
  ∀C.
      GCONTEXT C ⇒
      (∀l R.
          C P =label l⇒ R ⇒
          ∃C'.
              GCONTEXT C' ∧ R ⪰bis C' P ∧
              (WEAK_EQUIV ∘r (λx y. x =label l⇒ y)) (C Q)
                (C' Q)) ∧
```

```
    ∀R.
        C P =τ⇒ R ⇒
        ∃C'.
            GCONTEXT C' ∧ R ⪰_bis C' P ∧
            (WEAK_EQUIV ∘_r EPS) (C Q) (C' Q)
```

Traces are actually used in the proof of above lemma via the following “unfolding lemma”:

```
⊢ GCONTEXT C ∧ WGS E ∧ TRACE ((C ∘ FUNPOW E n) P) xs P' ∧
  LENGTH xs ≤ n ⇒
  ∃C'.
      GCONTEXT C' ∧ (P' = C' P) ∧
      ∀Q. TRACE ((C ∘ FUNPOW E n) Q) xs (C' Q)                 [unfolding_lemma4]
```

It roughly says, for any context C and weakly-guarded context 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 used for reasoning about the number of intermediate actions in weak transitions. For instance, from Def. 3.5, it is easy to see that, a weak transition either becomes shorter or remains the same when moving between $\succeq_{\mathrm{bis}}$-related processes. This property is essential in the proof of Lemma 3.9. We show only one such lemma, for the case of non-τ weak transitions passing into $\succeq_{\mathrm{bis}}$ (from left to right):

```
⊢ P ⪰_bis Q ⇒
  ∀xs l P'.
      TRACE P xs P' ∧ UNIQUE_LABEL (label l) xs ⇒
      ∃xs' Q'.
          TRACE Q xs' Q' ∧ P ⪰_bis Q ∧ LENGTH xs' ≤ LENGTH xs ∧
          UNIQUE_LABEL (label l) xs'                    [contracts_AND_TRACE_label]
```

With all above lemmas, we can thus finally prove Theorem 3.10:

```
⊢ WGS E ⇒ ∀P Q. P ⪰_bis E P ∧ Q ⪰_bis E Q ⇒ P ≈ Q
                                            [UNIQUE_SOLUTION_OF_CONTRACTIONS]
```

4.11. Unique solution of rooted contractions

The formal proof of “unique solution of rooted contractions theorem” (Theorem 3.14) 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 3.14 (having proved the precongruence of rooted contraction, we can use weakly-guarded expressions, without constraints on sums; that is, WG in place of WGS):

```
⊢ WG E ⇒ ∀P Q. P ⪰^c_bis E P ∧ Q ⪰^c_bis E Q ⇒ P ≈^c Q
                                     [UNIQUE_SOLUTION_OF_ROOTED_CONTRACTIONS]
```

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 context (WG) is required. In contrast, Milner’s “unique solution of equations” theorem for rooted bisimilarity ($\approx^{\mathrm{c}}$) has more rigid constraints, as the variables must be both strongly guarded and sequential.

5. Related work on formalisation

Monica Nesi did the first CCS formalisations for both pure and value-passing CCS [8, 22] using early versions of the HOL theorem prover.[7] Her main focus was on implementing decision procedures (as a ML

[7]Part of this work can now be found at https://github.com/binghe/HOL-CCS/tree/master/CCS-Nesi.

program, e.g. [26]) for automatically proving bisimilarities of CCS processes. Her work is the working basis of ours. However, the differences are substantial, especially in the way of defining bisimilarities. We greatly benefited from features and standard libraries in recent versions of HOL4, and our formalisation has covered a much larger spectrum of the (pure) CCS theory.

Bengtson, Parrow and Weber did a substantial formalisation work on CCS, π-calculi and ψ-calculi using Isabelle/HOL and its nominal logic, with the main focus on the handling of name binders [27, 28]. Other formalisations in this area include the early work of T. F. Melham [29] and O.A. Mohamed [30] in HOL, Compton [31] in Isabelle/HOL, Solange[8] in Coq and Chaudhuri et al. [32] in Abella, the latter focuses on 'bisimulation up-to' techniques (for strong bisimilarity) for CCS and π-calculus. Damien Pous [33] also formalised up-to techniques and some CCS examples in Coq. Formalisations less related to ours include Kahsai and Miculan [34] for the spi calculus in Isabelle/HOL, and Hirschkoff [35] for the π-calculus in Coq.

6. Conclusions and future work

In this paper, we have highlighted a formalisation of the theory of CCS in the HOL4 theorem prover. The formalisation supports 4 methods for establishing (strong and weak) bisimilarity results:

1. constructing a bisimulation (the standard bisimulation proof method);
2. constructing a 'bisimulation up-to';
3. employing algebraic laws;
4. defining a system of equations or contractions (i.e., the 'unique-solution' method)

The formalisation has actually 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. Indeed we have considered several 'unique-solution' results (for various equivalences and preorders); they share many parts of the proofs, but present a few delicate and subtle differences in a few points. In a paper-pencil proof, checking all details would be long and error-prune, since one is tempted to overlook details that tend to repeat themselves. A theorem-prover formalisation is in this respect helpful and reassuring. We think that the statements in the formalisation are easy to read and understand, as they are close to the original statements found in standard CCS textbooks [10, 2].

For the future work, it would be worth extending to multi-variable equations/contractions. A key aspect could be using unguarded constants as free variables (`FV`) and defining guardedness directly on expressions of type CCS (instead of CCS → CCS), then linking to contexts. For instance, an expression is weakly-guarded when each of its free variables, replaced by a hole, results in a weakly-guarded context:

```
weakly_guarded1 E ⇔
∀X. X ∈ FV E ⇒ ∀e. CONTEXT e ∧ (e (var X) = E) ⇒ WG e          [weakly_guarded1_def]
```

Formalising other equivalences and preorders could also be considered, notably the trace equivalences, as well as more refined process calculi such as value-passing CCS. On another research line, one could examine the formalisation of a different approach [36] to unique solutions, in which the use of contraction is replaced by semantic conditions on process transitions such as divergence.

Acknowledgements. We have benefitted from suggestions and comments from several people from the HOL community, including (in alphabet order) Robert Beers, Jeremy Dawson, Ramana Kumar, Michael Norrish, Konrad Slind, and Thomas Türk. The paper was written in memory of Michael J. C. Gordon, the creator of the HOL theorem prover.

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

References

[1] C. Tian, D. Sangiorgi, Unique Solutions of Contractions, CCS, and their HOL Formalisation, in: J. A. Pérez, S. Tini (Eds.), Proceedings Combined 25th International Workshop on Expressiveness in Concurrency and 15th Workshop on Structural Operational Semantics, Beijing, China, September 3, 2018, Vol. 276 of Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, 2018, pp. 122–139 (2018). `doi:10.4204/EPTCS.276.10`.

[2] R. Milner, Communication and concurrency, PHI Series in computer science, Prentice-Hall, 1989 (1989).

[3] J. C. M. Baeten, T. Basten, M. A. Reniers, Process Algebra: Equational Theories of Communicating Processes, Cambridge University Press, 2010 (2010). `doi:10.1017/CBO9781139195003`.

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

[5] A. W. Roscoe, Understanding Concurrent Systems, Springer, 2010 (2010). `doi:10.1007/978-1-84882-258-0`.

[6] D. Sangiorgi, Equations, contractions, and unique solutions, in: ACM SIGPLAN Notices, Vol. 50, ACM, 2015, pp. 421–432 (2015). `doi:10.1145/2676726.2676965`.
URL `https://hal.inria.fr/hal-01089205`

[7] D. Sangiorgi, Equations, contractions, and unique solutions, ACM Transactions on Computational Logic (TOCL) 18 (2017) 4 (2017). `doi:10.1145/2971339`.
URL `https://hal.inria.fr/hal-01647063`

[8] M. Nesi, A formalization of the process algebra CCS in high order logic, Tech. Rep. UCAM-CL-TR-278, University of Cambridge, Computer Laboratory (1992).
URL `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-278.pdf`

[9] D. Sangiorgi, R. Milner, The problem of "weak bisimulation up to", in: International Conference on Concurrency Theory, Springer, 1992, pp. 32–46 (1992). `doi:10.1007/BFb0084781`.
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.5964`

[10] R. Gorrieri, C. Versari, Introduction to Concurrency Theory, Transition Systems and CCS, Springer, Cham, 2015 (Sep. 2015). `doi:10.1007/978-3-319-21491-7`.

[11] R. J. van Glabbeek, A characterisation of weak bisimulation congruence, in: Processes, Terms and Cycles: Steps on the Road to Infinity, Springer, 2005, pp. 26–39 (2005). `doi:10.1007/11601548_4`.

[12] D. Sangiorgi, Introduction to Bisimulation and Coinduction, Cambridge University Press, 2011 (Oct. 2011). `doi:10.1017/CBO9780511777110`.

[13] S. Arun-Kumar, M. Hennessy, An efficiency preorder for processes, Acta Informatica 29 (8) (1992) 737–760 (1992). `doi:10.1007/BF01191894`.

[14] K. Slind, M. Norrish, A brief overview of hol4, in: International Conference on Theorem Proving in Higher Order Logics, Springer, 2008, pp. 28–32 (2008). `doi:10.1007/978-3-540-71067-7_6`.
URL `http://ts.data61.csiro.au/publications/nicta_full_text/1482.pdf`

[15] C. Tian, A Formalization of Unique Solutions of Equations in Process Algebra, Master's thesis, AlmaDigital, Bologna (Dec. 2017).
URL `http://amslaurea.unibo.it/14798/`

[16] HOL4 contributors, The HOL System LOGIC (Jun. 2018).
URL `http://sourceforge.net/projects/hol/files/hol/kananaskis-12/kananaskis-12-logic.pdf`

[17] M. J. C. Gordon, A. J. Milner, C. P. Wadsworth, Edinburgh LCF: A Mechanised Logic of Computation, Vol. 78 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 1979 (1979). `doi:10.1007/3-540-09724-4`.

[18] R. Milner, Logic for computable functions: description of a machine implementation, Tech. rep., Stanford Univ., Dept. of Computer Science (1972).
URL `http://www.dtic.mil/dtic/tr/fulltext/u2/785072.pdf`

[19] A. Church, A formulation of the simple theory of types, The journal of symbolic logic 5 (2) (1940) 56–68 (1940). `doi:10.2307/2266170`.

[20] J. Leslie-Hurd, The OpenTheory standard theory library, in: NASA Formal Methods Symposium, Springer, 2011, pp. 177–191 (2011). `doi:10.1007/978-3-642-20398-5_14`.

[21] R. Gorrieri, CCS (25, 12) is Turing-complete, Fundamenta Informaticae 154 (1-4) (2017) 145–166 (2017). `doi:10.3233/FI-2017-1557`.

[22] M. Nesi, Formalising a Value-Passing Calculus in HOL, Formal Aspects of Computing 11 (2) (1999) 160–199 (1999). `doi:10.1007/s001650050046`.

[23] HOL4 contributors, The HOL System DESCRIPTION (Jun. 2018).
URL `http://sourceforge.net/projects/hol/files/hol/kananaskis-12/kananaskis-12-description.pdf`

[24] M. Norrish, B. Huffman, Ordinals in HOL: Transfinite arithmetic up to (and beyond) ω_1, in: International Conference on Interactive Theorem Proving, Springer, 2013, pp. 133–146 (2013). `doi:10.1007/978-3-642-39634-2_12`.

[25] T. F. Melham, The HOL pred_sets Library, Universiy of Cambridge Computer Lab (feb 1992). `doi:10.1.1.219.5390`.
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.219.5390`

[26] R. Cleaveland, J. Parrow, B. Steffen, The concurrency workbench: A semantics-based tool for the verification of concurrent systems, ACM Transactions on Programming Languages and Systems (TOPLAS) 15 (1) (1993) 36–72 (1993). `doi:10.1145/151646.151648`.

[27] J. Bengtson, J. Parrow, A completeness proof for bisimulation in the pi-calculus using isabelle, Electronic Notes in Theoretical Computer Science 192 (1) (2007) 61–75 (2007). `doi:10.1016/j.entcs.2007.08.017`.

[28] J. Parrow, J. Bengtson, Formalising the pi-calculus using nominal logic, Logical Methods in Computer Science 5 (2009). `doi:10.2168/LMCS-5(2:16)2009`.

[29] T. F. Melham, A Mechanized Theory of the Pi-Calculus in HOL, Nord. J. Comput. 1 (1) (1994) 50–76 (1994). `doi:10.1.1.56.4370`.
URL `http://core.ac.uk/download/pdf/22878407.pdf`

[30] O. Aït Mohamed, Mechanizing a π-calculus equivalence in HOL, in: E. Thomas Schubert, P. J. Windley, J. Alves-Foss (Eds.), Higher Order Logic Theorem Proving and Its Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, pp. 1–16 (1995). `doi:10.1007/3-540-60275-5_53`.

[31] M. Compton, Embedding a fair CCS in Isabelle/HOL, in: Theorem Proving in Higher Order Logics: Emerging Trends Proceedings, 2005, p. 30 (2005). `doi:10.1.1.105.834`.
URL `https://web.comlab.ox.ac.uk/techreports/oucl/RR-05-02.pdf#page=36`

[32] K. Chaudhuri, M. Cimini, D. Miller, A lightweight formalization of the metatheory of bisimulation-up-to, in: Proceedings of the 2015 Conference on Certified Programs and Proofs, ACM, 2015, pp. 157–166 (2015). `doi:10.1145/2676724.2693170`.

[33] D. Pous, New up-to techniques for weak bisimulation, Theoretical Computer Science 380 (2007) 164–180 (2007). `doi:10.1016/j.tcs.2007.02.060`.

[34] T. Kahsai, M. Miculan, Implementing spi calculus using nominal techniques, in: Conference on Computability in Europe, Springer, 2008, pp. 294–305 (2008). `doi:10.1007/978-3-540-69407-6_33`.

[35] D. Hirschkoff, A full formalisation of π-calculus theory in the calculus of constructions, in: International Conference on Theorem Proving in Higher Order Logics, Springer, 1997, pp. 153–169 (1997). `doi:10.1007/BFb0028392`.

[36] A. Durier, D. Hirschkoff, D. Sangiorgi, Divergence and Unique Solution of Equations, in: R. Meyer, U. Nestmann (Eds.), 28th International Conference on Concurrency Theory (CONCUR 2017), Vol. 85 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017, pp. 11:1–11:16 (2017). `doi:10.4230/LIPIcs.CONCUR.2017.11`.
URL `http://drops.dagstuhl.de/opus/volltexte/2017/7784`