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

Chun Tian[a], Davide Sangiorgi[b]

[a] *University of Trento and Fondazione Bruno Kessler, Italy*
[b] *University of Bologna, Italy and INRIA, France*

Abstract

The unique solution of contractions is a proof technique for (weak) bisimilarity that overcomes certain syntactic limitations of Milner's "unique solution of equations" theorem. This paper presents an overview of a comprehensive formalisation of Milner's Calculus of Communicating Systems (CCS) in the HOL theorem prover (HOL4), with a focus towards the theory of unique solutions of equations and contractions. The formalisation consists of about 24,000 lines (1MB) of code in total. Some refinements of the "unique solution of contractions" theory itself are obtained. In particular we removed the constraints on summation, which must be guarded, by moving from contraction to *rooted contraction*. We prove the "unique solution of rooted contractions" theorem and show that rooted contraction is the coarsest precongruence contained in the contraction preorder.

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 [1], 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 [4, 5, 6].

Milner's unique-solution theorem for weak bisimilarity, however, has severe syntactic limitations: the equations must be both *guarded* and *sequential*. That is, variables of the equations can only occur underneath visible prefixes and summations. One way to overcome such limitations is to replace the equations with special inequations called *contractions* [7, 8]. Contraction is a preorder that, roughly, places some efficiency constraints on processes. The unique solution of contractions is defined as with equations: any two solutions must be componentwise bisimilar. The difference from 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 are eliminated. One constraint that still remains in [8] (where the issue is bypassed using a more restrictive CCS syntax) is the occurrences of sums (e.g. $P + Q$) due to the non-substitutivity of the contraction preorder in this case.

This paper presents a comprehensive formalisation of Milner's Calculus of Communicating Systems (CCS) in the HOL theorem prover (HOL4), with a focus towards the theory of unique solutions of equations and contractions. Many results in Milner's CCS book [1] are covered, since the unique-solution theorems rely on a large number of fundamental results. Indeed the formalisation encompasses all basic properties of strong and weak bisimilarities (e.g. the fixed-point and substitutivity properties), the basic properties of rooted bisimilarity and their algebraic laws. Further extensions include several versions of "bisimulation up to" techniques, the expansion and contraction preorder. Concerning rooted bisimilarity, the formalisation

[☆] This is an extended and refined version of the paper with the same title published in EXPRESS/SOS 2018 [3].

Email addresses: chun.tian@unitn.it (Chun Tian), davide.sangiorgi@unibo.it (Davide Sangiorgi)

[1] Part of this work was carried out when the first author was studying at the University of Bologna, Italy.

includes Hennessy's Lemma and Deng's Lemma, and two theorems saying that rooted bisimilarity is the coarsest (largest) congruence contained in weak bisimilarity ($\approx$): one is classical with the hypothesis that no process uses up all labels; the other without such hypothesis, essentially formalised van Glabbeek's proof [9]. Similar results are also proved for the rooted contraction preorder. In this respect, the work is also an extensive experiment using the HOL theorem prover with its recent developments, including its coinduction defining package.

From the point of view of the CCS theory, this formalisation has offered us the possibility of further refining the theory of unique solutions of contractions. In particular, the existing result [8] has 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. In this paper, the contraction-based proof technique is further refined by moving from the contraction preorder 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 use the sum operator.

Another benefit of the formalisation is that we can take advantage of results about different equivalences and preorders that share similar proof structures. Such structural simililarities can be found, for instance, in the following cases: the proofs 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 proofs about unique solution(s) 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); the proofs about various forms of enhancements of the bisimulation proof method (the 'up-to' techniques). In these cases, when 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.

We first consider the case of a single equation (or contraction), that we call the univariable case, then consider the multivariable case, where more than one equation (or contraction) with multiple equation variables is considered. The univariable theorems use λ-functions to present CCS equations, while the multivariable versions require a more careful and delicate treatment of CCS expressions with multiple variables and substitutions acting on them.

In contrast to modern literature, e.g. [2], we have followed Milner's original approach [10] and adopted the same type for both CCS equations and processes: those undefined constants in CCS terms are treated as (free) equation variables, and a CCS *process* is a CCS term without equation variables. This allows us a smoother move from the univariable case to the multivariable one. See Section 5 for details.

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 for the univariable case. Section 5 describes the extension to the multivariable case. Finally, Section 6 and 7 discuss related work, conclusions, and a few directions for future work.

2. CCS

Here we recall the core theory of CCS. We assume a possibly infinite set of names $\mathscr{L} = \{a, b, \ldots\}$ yielding input and $\overline{\text{output}}$ labels (actions), plus a special invisible action $\tau \notin \mathscr{L}$, and another possibly infinite set of agent variables $A, B, \ldots \in \mathscr{X}$. The class of CCS processes is then inductively defined from $\mathbf{0}$ (the terminated process) and agent variables by the operators of *prefixing* (.), *parallel composition* (|), *summation* (or *binary choice*, +), *restriction* (ν), *relabeling* ($[\cdot]$) and *recursion* (`rec`):

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

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

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

Figure 1: Structural Operational Semantics of CCS

In our presentation of CCS, the restriction operator takes a set of labels $L \subseteq \mathscr{L}$, rather than a single one. The relabeling operator takes a relabeling function $rf : \mathscr{L} \cup \overline{\mathscr{L}} \cup \{\tau\} \to \mathscr{L} \cup \overline{\mathscr{L}} \cup \{\tau\}$, that can handle multiple actions including τ. A valid relabeling function rf must however satisfy $rf(\tau) = \tau$ and $\forall l \in \mathscr{L} \cup \overline{\mathscr{L}}.\ rf(\bar{l}) = \overline{rf(l)}$ (with $\bar{\bar{l}} = l$ for all $l \in \mathscr{L}$). We sometimes omit a trailing $\mathbf{0}$, e.g., writing $a \mid b$ for $a.\,\mathbf{0} \mid b.\,\mathbf{0}$. A CCS process P may evolve to another one, say P', under an action μ, written by $P \xrightarrow{\mu} P'$. The transition semantics of CCS processes is given by means of a Labeled Transition System (LTS) expressed in Structural Operational Semantics (SOS) rules shown in Fig. 1. (The symmetric versions of the SOS rules for parallel composition and binary choice are omitted.) A CCS processes uses only *guarded sums* if all occurrences of sums are in the form $a.\,P + b.\,Q$. The *immediate derivatives* of a process P are 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: $\stackrel{\epsilon}{\Longrightarrow}$ is the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\stackrel{\mu}{\Longrightarrow}$ is $\stackrel{\epsilon}{\Longrightarrow}\xrightarrow{\mu}\stackrel{\epsilon}{\Longrightarrow}$ (the composition of the three relations). Moreover, $P \xrightarrow{\widehat{\mu}} P'$ holds if $P \xrightarrow{\mu} P'$ or $\mu = \tau \wedge P = P'$; similarly $P \stackrel{\widehat{\mu}}{\Longrightarrow} P'$ holds if $P \stackrel{\mu}{\Longrightarrow} P'$ or $\mu = \tau \wedge P = P'$). 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., $P\,\mathcal{R}\,Q$ means $(P, Q) \in \mathcal{R}$. 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., $\widetilde{P}\,\mathcal{R}\,\widetilde{Q}$ means $P_i\,\mathcal{R}\,Q_i$ for each index i of the tuples $\widetilde{P}$ and $\widetilde{Q}$. We use the symbol $\stackrel{\text{def}}{=}$ for abbreviations. For instance, $P \stackrel{\text{def}}{=} G$, where G is some expression, means that P stands for the expression G. 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 (weak)* bisimulation *if, whenever $P\,\mathcal{R}\,Q$,*

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

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

Strong bisimulation and strong bisimilarity ($\sim$) can be obtained by replacing the weak transition $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ in the above definition with normal transitions $Q \xrightarrow{\mu} Q'$ (similar for the other case). Weak bisimilarity is not preserved by the sum operator (except for guarded sums), i.e. $P_1 \approx Q_1$ and $P_2 \approx Q_2$ does not imply $P_1 + P_2 \approx Q_1 + Q_2$. For this, Milner introduced the concept of *observational congruence*, also called *rooted bisimilarity* [2, 11]:

Definition 2.2. *Two processes P and Q are* rooted bisimilar*, written as $P \approx^{\text{c}} Q$, if*

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

The above definition also brings up a proof technique for proving rooted bisimilarity from a given bisimulation. The next lemma plays an important role in proving some key results in this paper:

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

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

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

One basic property of rooted bisimiarity is congruence, indeed the coarsest congruence contained in bisimilarity (c.f. Section 4.7 for more details):

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

3. Equations and contractions

In the CCS syntax, a recursion $\mathtt{rec}\, A.\, P$ acts as a binder for A in the body P. This gives rise, in the expected manner, to the notions of *free* and *bound* recursion variables in a CCS expression. For instance, X is free in $a.\, X + \mathtt{rec}\, Y.\, (b.\, Y)$ while Y is bound; in $a.\, X + \mathtt{rec}\, X.\, (b.\, X)$, X is both free and bound. A tem without free variables is a *process*.

In this paper (and the formalisation work), we use the agent variables also as *equation variables*. This eliminates the need of another type for equations, and we can reuse the existing variable substitution operation (c.f. SOS rule for the Recursion in Fig. 1) for the substitution of equation variables. For example, the result of substituting variable X with $\mathbf{0}$ in $a.\, X + \mathtt{rec}\, X.\, (b.\, X)$, written $(a.\, X + \mathtt{rec}\, X.\, (b.\, X))\{\mathbf{0}/X\}$, is $a.\, \mathbf{0} + \mathtt{rec}\, X.\, (b.\, X)$ (with the part $\mathtt{rec}\, X.\, (b.\, X)$ untouched). Multivariable substitutions are written in the same syntax, e.g. $E\{\widetilde{P}/\widetilde{X}\}$. Whenever $\widetilde{X}$ is clear from the context, we may also write $E[\widetilde{P}]$ instead of $E\{\widetilde{P}/\widetilde{X}\}$ (and $E[P]$ for $E\{P/X\}$ if there is a single equation variable X.)

3.1. Systems of equations

When discussing equations it is standard to talk about 'context'. This is a a CCS expression possibly containing free variables that, however, may not occur within the body of recursive definitions. Milner's "unique solution of equations" theorems [1] intuitively say 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.

Definition 3.1 (equations)**.** *Assume that, for each* i *of a countable indexing set* I*, we have variables* X_i*, and expressions* E_i *possibly containing such variables* $\cup_i \{X_i\}$*. Then* $\{X_i = E_i\}_{i \in I}$ *is a* system of equations*. (There is one equation* E_i *for each variable* X_i*.)*

The components of $\widetilde{P}$ need not be different from each other, as it must be for the variables $\widetilde{X}$. If the system has infinitely many equations, the tuples $\widetilde{P}$ and $\widetilde{X}$ are infinite too.

Definition 3.2 (solutions and unique solutions)**.** *Suppose* $\{X_i = E_i\}_{i \in I}$ *is a system of equations:*

- $\widetilde{P}$ *is a* solution of the system of equations (for $\approx$) *if for each* i *it holds that* $P_i \approx E_i[\widetilde{P}]$*;*
- *The system has* a unique solution for $\approx$ *if whenever* $\widetilde{P}$ *and* $\widetilde{Q}$ *are both solutions then* $\widetilde{P} \approx \widetilde{Q}$*.*

Similarily, the *(unique) solution of a system of equations for* $\sim$ (or for $\approx^{\mathrm{c}}$) can be obtained by replacing all occurrences of $\approx$ in above definition with $\sim$ and $\approx^{\mathrm{c}}$, respectively.

For instance, the solution of the equation $X = a.\,X$ is the process $R \stackrel{\mathrm{def}}{=} \mathtt{rec}\,A.\ (a.\,A)$, and for any other solution P we have $P \approx R$. In contrast, the equation $X = a \mid X$ has solutions that may be quite different, for instance, K and $K \mid b$, for $K \stackrel{\mathrm{def}}{=} \mathtt{rec}\,K.\ (a.\,K)$. (Actually any process capable of continuously performing a–actions is a solution of $X = a \mid X$.)

Examples of systems that do not have unique solutions are: $X = X$, $X = \tau.\,X$ and $X = a \mid X$.

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 each* X_i *is underneath a prefix;*
- guarded *if, in each* E_i*, each occurrence of each* X_i *is underneath a* visible *prefix;*
- sequential *if, in each* E_i*, each occurrence of each* X_i *is only underneath prefixes and sums.*

In other words, if a system of equations is sequential, then for each E_i, any subexpression of E_i in which X_j appears, apart from X_j itself, is a sum of prefixed expressions. For instance,

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

Milner has three versions of "unique solution of equations" theorems, for $\sim$, $\approx$ and $\approx^{\mathrm{c}}$, respectively, though only the following two versions are explicitly mentioned in [1, p. 103, 158]:

Theorem 3.4 (unique solution of equations for $\sim$)**.** *Let* E_i *be weakly guarded with free variables in* $\widetilde{X}$*, and let* $\widetilde{P} \sim \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \sim \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$*. Then* $\widetilde{P} \sim \widetilde{Q}$*.*

Theorem 3.5 (unique solution of equations for $\approx^{\mathrm{c}}$)**.** *Let* E_i *be guarded and sequential with free variables up to* $\widetilde{X}$*, and let* $\widetilde{P} \approx^{\mathrm{c}} \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \approx^{\mathrm{c}} \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$*. Then* $\widetilde{P} \approx^{\mathrm{c}} \widetilde{Q}$*.*

The version of Milner's unique-solution theorem for $\approx$ further requires that all sums are guarded:

Theorem 3.6 (unique solution of equations for $\approx$)**.** *Let* E_i *be guarded and sequential with free variables up to* $\widetilde{X}$*, and let* $\widetilde{P} \approx \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \approx \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$*. Then* $\widetilde{P} \approx \widetilde{Q}$*.*

The proof of the theorem above exploits an invariance property on immediate derivatives of guarded and sequential expressions, and then extracts a bisimulation (up to bisimilarity) out of the solutions of the system. To see the need of the sequentiality condition, consider the equation (from [1]) $X = \boldsymbol{\nu} a\,(a.\,X \mid \overline{a})$ where X is guarded but not sequential. Any process that does not use a is a solution, e.g. $\mathbf{0}$ and $b.\,\mathbf{0}$.

For more details of above three theorems, see Section 4.10 for the univariable case and Section 5 for the multivariable case.

3.2. Expansions and Contractions

Milner's "unique solution of equations" theorem for $\approx$ (Theorem 3.6) brings a new proof technique for proving (weak) bisimilarities. However, it has limitations: the equations must be guarded and sequential (moreover all sums where equation variables appear must be guarded sums). This limits the usefulness of the technique, since the occurrences of other operators above the variables, such as parallel composition and restriction, in general cannot be removed. The constraints of Theorem 3.6, however, can be weakened if we move from equations to a special kind of inequations called *contractions*.

Intuitively, the bisimilarity contraction $\succeq_{\mathrm{bis}}$ is a preorder such that, $P \succeq_{\mathrm{bis}} Q$ holds if $P \approx Q$ and, in addition, Q *has the possibility of being at least as efficient as* P (as far as τ-actions are performed). Process Q, however, may be non-deterministic and may have other ways of doing the same work, and these could be slow (i.e., involving more τ-actions than those performed by P).

Definition 3.7 (contraction)**.** *A process relation $\mathcal{R}$ is a* (bisimulation) contraction *if, whenever $P\,\mathcal{R}\,Q$,*

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

Two processes P and Q are in the bisimilarity contraction*, written as $P \succeq_{\mathrm{bis}} Q$ (P* contracts to *Q), if $P\,\mathcal{R}\,Q$ for some contraction $\mathcal{R}$. Sometimes we write $\preceq_{\mathrm{bis}}$ for the inverse of $\succeq_{\mathrm{bis}}$.*

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 3.7, on the challenges from Q, entirely ignores efficiency: it is the same clause of weak bisimulation —the final derivatives are indeed required to be related by $\approx$, rather than by $\mathcal{R}$.

Bisimilarity contraction is coarser than the *expansion relation* $\succeq_{\mathrm{e}}$ [12, 7]:

Definition 3.8. *A process relation $\mathcal{R}$ is an* expansion *if, whenever we have $P\,\mathcal{R}\,Q$, for all μ*

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

Two processes P and Q are in the bisimilarity expansion*, written as $P \succeq_{\mathrm{e}} Q$ (P* expands *Q), if $P\,\mathcal{R}\,Q$ for some expansion $\mathcal{R}$.*

The expansion preorder is widely used in proof techniques for bisimilarity and intuitively refines bisimilarity by formalising the idea of "efficiency" between processes. Clause (1) is the same in the both preorders, while in clause (2) expansion uses $P \stackrel{\mu}{\Longrightarrow} P'$, rather than $P \stackrel{\widehat{\mu}}{\Longrightarrow} P'$; moreover in clause (2) of Def. 3.7 the final derivatives are simply required to be bisimilar ($P' \approx Q'$). Intuitively, $P \succeq_{\mathrm{e}} Q$ holds if $P \approx Q$ and, in addition, *Q is always at least as efficient as P*.

Example 3.9. *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 $\succeq_{\mathrm{e}} \subset \succeq_{\mathrm{bis}}$.*

In the same way as bisimilarity, both the expansion and the contraction preorders are preserved by all CCS operators except the summation. The proofs are similar, see e.g. [8].

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 $=$ in Def. 3.1. Thus a system of contractions is a set $\{X_i \succeq E_i\}_{i\in I}$ where I is an indexing set and each E_i contains variables up to $\widetilde{X}$.

Now we recall the "unique solution of contractions" theorem [8], which weakens the requirements of Milner's result (Theorem 3.6).

Lemma 3.10. *Suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\succeq_{\mathrm{bis}}$ of a system of weakly-guarded contractions that uses guarded sums. For any context C that uses guarded sums, if $C[\widetilde{P}] \stackrel{\mu}{\Longrightarrow} R$, then there is a context C' that uses guarded sums such that $R \succeq_{\mathrm{bis}} C'[\widetilde{P}]$ and $C[\widetilde{Q}] \stackrel{\widehat{\mu}}{\Longrightarrow} R' \approx C'[\widetilde{Q}]$ for some R'.*

Proof. (sketch from [8]) Let n be the length (the number of one-step transitions) of a transition $C[\widetilde{P}] \stackrel{\mu}{\Longrightarrow} R$, and let $C''[\widetilde{P}]$ and $C''[\widetilde{Q}]$ be the processes obtained from $C[\widetilde{P}]$ and $C[\widetilde{Q}]$ by unfolding the definitions of the contractions n times. Thus in C'' each hole is underneath at least n prefixes, and cannot contribute to an action in the first n transitions; moreover all the contexts use guarded sums.

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

Theorem 3.11 (unique solution of contractions [8])**.** *Let E_i be weakly guarded (and with guarded sums only) with free variables in $\widetilde{X}$, and let $\widetilde{P} \succeq_{\text{bis}} \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \succeq_{\text{bis}} \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$. Then $\widetilde{P} \approx \widetilde{Q}$.*

Proof. We prove $P_i \approx Q_i$ (for each $i \in I$) by considering the following relation

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

Obviously we have $(P_i, Q_i) \in \mathcal{R}$ (by taking $C = E_i$, and the fact that $\succeq_{\text{bis}}$ implies $\approx$). It remains to show that $\mathcal{R}$ is a bisimulation. Suppose $(R, S) \in \mathcal{R}$ via a context C. For any R' such that $R \xrightarrow{\mu} R'$, we have to find a S' with $S \stackrel{\widehat{\mu}}{\Longrightarrow} S'$ and $(R', S') \in \mathcal{R}$. From $R \approx C[\widetilde{P}]$, we derive $C[\widetilde{P}] \stackrel{\widehat{\mu}}{\Longrightarrow} R'' \approx R'$ for some R''. Then by Lemma 3.10, there exists C' with $R'' \succeq_{\text{bis}} C'[\widetilde{P}]$ and $C[\widetilde{Q}] \stackrel{\widehat{\mu}}{\Longrightarrow} S'' \approx C'[\widetilde{Q}]$ for some S''. From $S \approx C[\widetilde{Q}]$ and $C[\widetilde{Q}] \stackrel{\widehat{\mu}}{\Longrightarrow} S''$, by induction and definition of $\approx$, we find S' with $S \stackrel{\widehat{\mu}}{\Longrightarrow} S'$. This completes the proof, as we have $R' \approx C'[\widetilde{P}]$ and $S' \approx C'[\widetilde{Q}]$ by transitivity of $\approx$ and the fact that $\succeq_{\text{bis}}$ implies $\approx$. (The other side from S follows in the same manner.) See Fig. 2 for a visual illustration. □

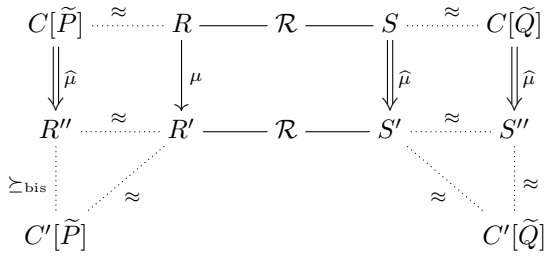

Figure 2: Proof illustration of Theorem 3.11 (showing $\mathcal{R}$ is a bisimulation)

3.4. Rooted contraction

Theorem 3.11 brings a new proof technique for bisimilarity, which is much less restrictive than Milner's Theorem 3.6 (but with the additional costs of checking $\succeq_{\text{bis}}$ in addition to $\approx$). However, comparing with Milner's Theorem 3.4, there is a constraint: the involved CCS expression may only use guarded sums. The is mainly due to the fact that $\approx$ is not a congruence and also $\succeq_{\text{bis}}$ is not a precongruence. Inspired by rooted bisimilarity, to eliminate the restriction on guarded sums, we have extended the idea of contractions by moving to *rooted contractions*:

Definition 3.12. *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 $P \stackrel{\mu}{\Longrightarrow} P'$ and $P' \approx Q'$.*

The above definition was found with the help of interactive theorem proving. The following two principles were adopted when searching for a possible definition: (1) the definition should not be coinductive, along the lines of rooted bisimilarity $\approx^{\text{c}}$ (Def. 2.2); (2) the definition should be built on top of the existing *contraction* relation $\succeq_{\text{bis}}$. Furthermore, we needed to show that the definition found is the best possible one, by showing that it yields the coarsest precongruence contains in $\succeq_{\text{bis}}$; the proof is similar to the analogous result for $\approx^{\text{c}}$. See Section 4.7 for more details.

Theorem 3.13. *$\succeq^{\text{c}}_{\text{bis}}$ is a precongruence in CCS, and it is the coarsest precongruence contained in $\succeq_{\text{bis}}$.*

For this new relation, the analogous of Lemma 3.10 and of Theorem 3.11 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 3.14. *Suppose $\widetilde{P}$ and $\widetilde{Q}$ are solutions for $\succeq^{\mathrm{c}}_{\mathrm{bis}}$ of a system of weakly-guarded contractions. For any context C, if $C[\widetilde{P}] \stackrel{\mu}{\Longrightarrow} R$, then there is a context C' such that $R \succeq_{\mathrm{bis}} C'[\widetilde{P}]$ and $C[\widetilde{Q}] \stackrel{\mu}{\Longrightarrow} R' \approx C'[\widetilde{Q}]$ for some R'.*

The next lemma is actually Lemma 3.13 of [1, p. 102], and is needed to prove Milner's "unique solution of equations for $\sim$" (Theorem 3.4):

Lemma 3.15. *If the variables $\widetilde{X}$ are weakly guarded in E, and $E\{\widetilde{P}/\widetilde{X}\} \xrightarrow{\alpha} P'$, then P' takes the form $E'\{\widetilde{P}/\widetilde{X}\}$ (for some expression E'), and moreover, for any $\widetilde{Q}$, $E\{\widetilde{Q}/\widetilde{X}\} \xrightarrow{\alpha} E'\{\widetilde{Q}/\widetilde{X}\}$.*

Theorem 3.16 (Unique solution of rooted contractions)**.** *Let E_i be weakly guarded with free variables in $\widetilde{X}$, and let $\widetilde{P} \succeq^{\mathrm{c}}_{\mathrm{bis}} \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$, $\widetilde{Q} \succeq^{\mathrm{c}}_{\mathrm{bis}} \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}$. Then $\widetilde{P} \approx^{\mathrm{c}} \widetilde{Q}$.*

Proof. We prove $P_i \approx^{\mathrm{c}} Q_i$ (for each $i \in I$) by considering the following relation

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

Following the same steps in the proof of Theorem 3.11 (using Lemma 3.14 in place of Lemma 3.10), we can prove that $(P_i, Q_i) \in \mathcal{R}$ and $\mathcal{R}$ is indeed a bisimulation. But this only shows $P_i \approx Q_i$. To further show $P_i \approx^{\mathrm{c}} Q_i$, we appeal to Lemma 2.3: for any P' such that $P_i \stackrel{\mu}{\longrightarrow} P'$, we have to find a Q' with $Q_i \stackrel{\mu}{\Longrightarrow} Q'$ and $(R', S') \in \mathcal{R}$. From $P_i \succeq^{\mathrm{c}}_{\mathrm{bis}} E_i[\widetilde{P}]$, by definition of $\succeq^{\mathrm{c}}_{\mathrm{bis}}$ we derive $E_i[\widetilde{P}] \stackrel{\mu}{\longrightarrow} P''$ for some P''. Then by Lemma 3.15 there exists a context E' with $P'' = E'[\widetilde{P}]$ and $E_i[\widetilde{Q}] \stackrel{\mu}{\longrightarrow} E'[\widetilde{Q}]$. From $Q_i \succeq^{\mathrm{c}}_{\mathrm{bis}} E_i[\widetilde{Q}]$ and $E_i[\widetilde{Q}] \stackrel{\mu}{\longrightarrow} E'[\widetilde{Q}]$, by definition of $\succeq^{\mathrm{c}}_{\mathrm{bis}}$ we have $Q_i \stackrel{\mu}{\Longrightarrow} Q'$ for some Q' and $Q' \approx E'[\widetilde{Q}]$. This completes the proof, as we have $P' \approx C[\widetilde{P}]$ (by the fact that $\succeq_{\mathrm{bis}}$ implies $\approx$) and $Q' \approx C[\widetilde{Q}]$. (The other side from Q_i follows in the same manner.) See Fig. 3 for a visual illustration. □

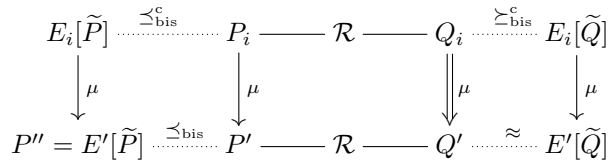

Figure 3: Proof illustration of Theorem 3.16 (showing $P_i \approx^{\mathrm{c}} Q_i$)

4. The formalisation

We highlight here a formalisation of CCS in the HOL theorem prover (HOL4) [13, 14], with a focus towards the theory (and formal proofs) of the unique solution of equations/contractions theorems mentioned in Section 3 and 3.2. All proof scripts are available as part of HOL's official examples[2]. The work so far consists of about 24,000 lines (1MB) of code in total, in which about 5,000 lines were derived from the early work of Monica Nesi [15] on HOL88, with major modifications.

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, while some other theorem provers in HOL family (e.g. Isabelle/HOL) have certain extensions. Indeed the HOL has considerably simpler logical foundations than most other theorem provers. As a consequence, theories and proofs verified in HOL are easier to understand to people who are not familar with more advanced dependent type theories.

HOL4 is implemented in Standard ML, the same programming language who plays three different roles:

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

1. the underlying implementation language for the core HOL engine;
2. the language in which proof tactics are implemented;
3. the interface language of the HOL proof scripts and 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.

4.1. Higher Order Logic

HOL is a formal system of typed logical terms. The types are expressions that denote sets (in the universe $\mathcal{U}$). HOL's type system is much simpler than those based on dependent types and other type theories. There are four kinds of types in the HOL logic, as illustrated in Fig. 4 for its BNF grammar. Noticed that, in HOL the standard atomic types *bool* and *ind* denote, respectively, the distinguished two-element set 2 and the distinguished infinite set I.

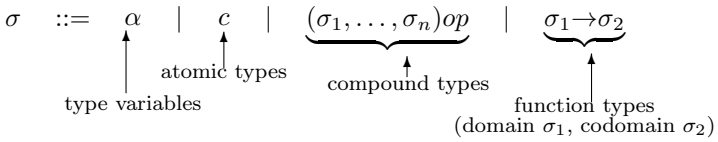

Figure 4: HOL's type grammar

The terms of the HOL logic are expressions that denote elements of the sets denoted by types. There're four kinds of terms in the HOL logic. There can be described approximately by the BNF grammar in Fig. 5.

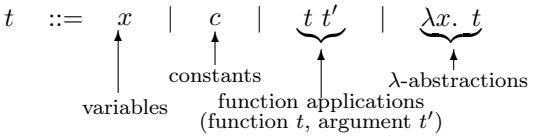

Figure 5: HOL's term grammar

The deductive system of the HOL is specified by eight primitive derivative rules: (c.f. [16] for more details.)

1. Assumption introduction (`ASSUME`);
2. Reflexivity (`REFL`);
3. β-conversion (`BETA_CONV`);
4. Substitution (`SUBST`);
5. Abstraction (`ABS`);
6. Type instantiation (`INST_TYPE`);
7. Discharging an assumption (`DISCH`);
8. Modus Ponens (`MP`).

All proofs are eventually reduced to applications of the above rules, which also give the semantics of two foundamental logical connectives, the equality ($=$) and implication ($\Rightarrow$). The rest logical connectives and first-order quantifiers, including the logical true (`T`) and false (`F`), are further defined as λ-functions: (one

rarely needs to care things at this level, however.)

$$\begin{aligned}
&\vdash \mathrm{T} = ((\lambda x_{bool}.\ x) = (\lambda x_{bool}.\ x))\\
&\vdash \forall = \lambda P_{\alpha\to bool}.\ P = (\lambda x.\ \mathrm{T})\\
&\vdash \exists = \lambda P_{\alpha\to bool}.\ P(\varepsilon\, P)\\
&\vdash \mathrm{F} = \forall b_{bool}.\ b\\
&\vdash \neg = \lambda b.\ b \Rightarrow \mathrm{F}\\
&\vdash \wedge = \lambda b_1\ b_2.\ \forall b.\ (b_1 \Rightarrow (b_2 \Rightarrow b)) \Rightarrow b\\
&\vdash \vee = \lambda b_1\ b_2.\ \forall b.\ (b_1 \Rightarrow b) \Rightarrow ((b_2 \Rightarrow b) \Rightarrow b)\\
&\vdash \mathrm{One_One} = \lambda f_{\alpha\to\beta}.\ \forall x_1\ x_2.\ (f\ x_1 = f\ x_2) \Rightarrow (x_1 = x_2)\\
&\vdash \mathrm{Onto} = \lambda f_{\alpha\to\beta}.\ \forall y.\ \exists x.\ y = f\ x\\
&\vdash \mathrm{Type_Definition} = \lambda P_{\alpha\to bool}\ rep_{\beta\to\alpha}.\ \mathrm{One_One}\ rep\ \wedge\ (\forall x.\ Px = (\exists y.\ x = rep\ y))
\end{aligned}$$

It deserves to mention that, the last logical constant above, Type_Definition, can be used to define new HOL types as bijections of subsets of existing types. (c.f. [20] for more details.) HOL's `Datatype` package [21, 22] automates this tedious process, and can be used to define needed types in CCS.

Finally, the whole HOL *standard* theory is based on the following four axioms, from which the almost[3] entire mathematics can be formalised:

`BOOL_CASES_AX`	$\vdash \forall b.\ (b = \mathrm{T})\ \vee\ (b = \mathrm{F})$
`ETA_AX`	$\vdash \forall f_{\alpha\to\beta}.\ (\lambda x.\ f\ x) = f$
`SELECT_AX`	$\vdash \forall P_{\alpha\to bool}\ x.\ P\ x \Rightarrow P(\varepsilon\ P)$
`INFINITY_AX`	$\vdash \exists f_{ind\to ind}.\ \mathrm{One\_One}\ f\ \wedge\ \neg(\mathrm{Onto}\ f)$

Notice that, usually the above four axioms are the only axioms allowed in conventional formalisation projects in HOL: adding new axioms may either break the logic consistencies or make the whole work less convincible. The theorem prover (in general) guarantees that, without adding new axioms, if both the definitions and the statements of theorems are correct, then any proof which successfully completes must be correct. This is also the case here: SOS rules of CCS are *not* axioms but consequences of an inductive relation.

4.2. CCS: the formal theory

The CCS formalisation starts with type definitions for action, relabeling and the CCS term itself. We use the type "β `Label`" (`'b` or β is a type variable) for all visible actions. The cardinality and the exact elements depend on the type variable β supplied by the end user. All actions are divided into input and $\overline{\text{output}}$ ones. This type is defined by HOL's `Datatype` package in the following syntax:

```
Datatype: Label = name 'b | coname 'b
End
```

For instance, if the type β is instantiated by the type `string`, then HOL terms `name` "a" and `coname` "b" denote the input action a and output action $\overline{\mathrm{b}}$, respectively. The type "β `Action`" is the union of all visible actions, plus the invisible action τ (`tau`). For instace, the input action a and output action $\overline{\mathrm{b}}$ of type "β `Action`" are denoted by `label` (`name` "a") and `label` (`coname` "b"), respectively.

The type "(α, β) `CCS`", accounting for the CCS expressions, where `'a` (or α) is the type variable[4] for agent variables, is then defined inductively: (β `Relabeling` is the type of all relabeling functions. They (and related operations) are not discussed in this paper.)

[3] HOL is strictly weaker than ZFC (the Zermelo-Frankel set theory with the Axiom of Choice), thus not all theorems valid in ZFC can be formalised in HOL. (c.f. [16] for more details.)

[4] The order of type variables α and β is arbitrary. Our choice is aligned with the literature. For instance, CCS(h, k) is the CCS subcalculus that can use at most h constants and k actions [23]. To formalize theorems on such CCS(h, k), the required 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.

```
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
End
```

The above definitions allow us to use terms like `nil` and `sum P Q` in HOL. Their correspondences with conventional CCS notations in the literature, are given in Table 1, where most CCS operators have also the more readable abbreviation forms, either for end user or for TEX outputs. (All formal theorems and definitions in this paper are generated from HOL4. Also, by default, all theorems are fully specialised, removing outermost universal quantifiers.)

CCS concept	Notation	HOL term	HOL abbrev.	TEX outputs
nil	$\mathbf{0}$	`nil`	`nil`	$\mathbf{0}$
prefix	$\mu.P$	`prefix u P`	`u..P`	$u.P$
summation	$P+Q$	`sum P Q`	`P + Q`	$P + Q$
parallel composition	$P \mid Q$	`par P Q`	`P || Q`	$P \mid Q$
restriction	$(\nu L)\ P$	`restr L P`	`(nu L) P`	$(\nu L)\ P$
recursion	$\mathtt{rec}\ A.\ P$	`rec A P`	`rec A P`	`rec` $A\ P$
relabeling	$P\ [rf]$	`relab P rf`	`relab P rf`	`relab` $P\ rf$
constant/variable	A	`var A`	`var A`	`var` A
invisible action	τ	`tau`	`tau`	τ
input action	a	`label (name a)`	`In(a)`	$\mathrm{in}\ a$
output action	$\overline{a}$	`label (coname a)`	`Out(a)`	$\overline{\mathrm{out}}\ a$
variable substitution	$E\{E'/X\}$	`CCS_Subst E E' X`	`[E'/X] E`	$[E'/X]\ E$
transition	$P \xrightarrow{\mu} Q$	`TRANS P u Q`	`P --u-> Q`	$P\ {-u}\!\rightarrow\ Q$
weak transition	$P \stackrel{\mu}{\Longrightarrow} Q$	`WEAK_TRANS P u Q`	`P ==u=> Q`	$P\ {=u}\!\Rightarrow\ Q$
ϵ–transition	$P \stackrel{\epsilon}{\Longrightarrow} Q$	`EPS P Q`	`EPS P Q`	$P \stackrel{\epsilon}{\Longrightarrow} Q$

Table 1: Syntax of some CCS concepts in HOL

The transition semantics of CCS processes strictly follows the SOS rules given in Fig. 1. However, they are not axioms, but the consequence of an *inductive relation* definition of `TRANS` by HOL's `HOL_reln` function. (c.f. [22, p. 219] for more details.) A successful invocation of the definitional principle returns three important theorems (*rules*, *ind* and *cases*):

- *rules* is a conjunction of implications that will be the same as the input term. In fact, the following formal versions of SOS rules are extracted from the *rules* theorem:

$$\vdash\ u.E\ {-u}\!\rightarrow\ E \qquad \texttt{[PREFIX]}$$
$$\vdash\ E\ {-u}\!\rightarrow\ E_1 \implies E\ +\ E'\ {-u}\!\rightarrow\ E_1 \qquad \texttt{[SUM1]}$$
$$\vdash\ E\ {-u}\!\rightarrow\ E_1 \implies E'\ +\ E\ {-u}\!\rightarrow\ E_1 \qquad \texttt{[SUM2]}$$
$$\vdash\ E\ {-u}\!\rightarrow\ E_1 \implies E\ \mid\ E'\ {-u}\!\rightarrow\ E_1\ \mid\ E' \qquad \texttt{[PAR1]}$$
$$\vdash\ E\ {-u}\!\rightarrow\ E_1 \implies E'\ \mid\ E\ {-u}\!\rightarrow\ E'\ \mid\ E_1 \qquad \texttt{[PAR2]}$$

$$\frac{E\ {-\texttt{label}\ l}\!\rightarrow\ E_1 \quad E'\ {-\texttt{label}\ (\texttt{COMPL}\ l)}\!\rightarrow\ E_2}{E\ \mid\ E'\ {-\tau}\!\rightarrow\ E_1\ \mid\ E_2}\ \textsf{PAR3}$$

$$\frac{E\ {-u}\!\rightarrow\ E' \quad u = \tau \vee u = \texttt{label}\ l \wedge l \notin L \wedge \texttt{COMPL}\ l \notin L}{(\nu L)\ E\ {-u}\!\rightarrow\ (\nu L)\ E'}\ \textsf{RESTR}$$

$$\frac{E\ -u\rightarrow\ E'}{\texttt{relab}\ E\ rf\ -\texttt{relabel}\ rf\ u\rightarrow\ \texttt{relab}\ E'\ rf}\ \text{RELABELING}$$

$$\frac{[\texttt{rec}\ X\ E/X]\ E\ -u\rightarrow\ E_1}{\texttt{rec}\ X\ E\ -u\rightarrow\ E_1}\ \text{REC}$$

- *ind* is the induction principle for the relation (see Section 5 for the exact form and its application).
- *cases* is the so-called 'cases' or 'inversion' theorem for the relations, and is used to decompose an element in the relation into the possible ways of obtaining it by the rules. For instance, it leads to the following theorem:

$$\vdash\ E\ +\ E'\ -u\rightarrow\ E''\ \Longleftrightarrow\ E\ -u\rightarrow\ E''\ \vee\ E'\ -u\rightarrow\ E''\qquad\texttt{[TRANS\_SUM\_EQ]}$$

The last SOS rule `REC` (Recursion) says that if we substitute all occurrences of the variable A in P to $(\texttt{rec}\ A.\,P)$ and the resulting process has a transition to P' with action u, then $(\texttt{rec}\ A.\,P)$ has the same transition. Here, "$[\texttt{rec}\ X\ E/X]\ E$" is an abbreviation for "`CCS_Subst` E $(\texttt{rec}\ X\ E)$ X", where `CCS_Subst` is a recursive function substituting all appearences of a free variable into another CCS term. For most CCS operators `CCS_Subst` just recursively goes into a deeper level without changing anything, e.g.

$$\vdash\ [E'/X]\ (E_1\ +\ E_2)\ =\ [E'/X]\ E_1\ +\ [E'/X]\ E_2\qquad\texttt{[CCS\_Subst\_sum]}$$

The only two insteresting cases are at agent variables and recursion:

$$\vdash\ [E'/X]\ (\texttt{var}\ Y)\ =\ \textbf{if}\ Y\ =\ X\ \textbf{then}\ E'\ \textbf{else}\ \texttt{var}\ Y\qquad\texttt{[CCS\_Subst\_var]}$$
$$\vdash\ [E'/X]\ (\texttt{rec}\ Y\ E)\ =\ \textbf{if}\ Y\ =\ X\ \textbf{then}\ \texttt{rec}\ Y\ E\ \textbf{else}\ \texttt{rec}\ Y\ ([E'/X]\ E)\qquad\texttt{[CCS\_Subst\_rec]}$$

The variable substitutions only affects free variables: the variable Y in "$\texttt{rec}\ Y\ E$" is bound and therefore the substitution ignores it. `CCS_Subst`

A useful facility exploiting the interplay between HOL4 and Standard ML (that follows an idea by Nesi [15]) is a complex Standard ML function taking a CCS process and returning a theorem indicating all direct transitions of the process.[5] For instance, we know that the process $(a.\,\mathbf{0}\mid\bar{a}.\,\mathbf{0})$ has three possible transitions: $(a.\,\mathbf{0}\mid\bar{a}.\,\mathbf{0})\xrightarrow{a}(\mathbf{0}\mid\bar{a}.\,\mathbf{0})$, $(a.\,\mathbf{0}\mid\bar{a}.\,\mathbf{0})\xrightarrow{\bar{a}}(a.\,\mathbf{0}\mid\mathbf{0})$ and $(a.\,\mathbf{0}\mid\bar{a}.\,\mathbf{0})\xrightarrow{\tau}(\mathbf{0}\mid\mathbf{0})$. To completely describe all possible transitions of a process, if done manually, the following two facts must be proved: (1) there are indeed the three transitions mentioned above; (2) there are no other transitions. For large CCS processes it is surprisingly tedious to manually derive all the possible transitions and prove the non-existence of others. This shows the usefulness of appealing to an ML function `CCS_TRANS_CONV` that is designed to automate the whole process. For instance, calling it on $(a.\,\mathbf{0}\mid\bar{a}.\,\mathbf{0})$ the function returns the following theorem which describes all the transitions (just one-step) of the process:

$$\begin{array}{l}\vdash\ \text{in "a"}.\mathbf{0}\ \mid\ \overline{\text{out}}\ \text{"a"}.\mathbf{0}\ -u\rightarrow\ E\ \Longleftrightarrow\\ \quad(u\ =\ \text{in "a"}\ \wedge\ E\ =\ \mathbf{0}\ \mid\ \overline{\text{out}}\ \text{"a"}.\mathbf{0}\ \vee\ u\ =\ \overline{\text{out}}\ \text{"a"}\ \wedge\ E\ =\ \text{in "a"}.\mathbf{0}\ \mid\ \mathbf{0})\ \vee\\ \quad u\ =\ \tau\ \wedge\ E\ =\ \mathbf{0}\ \mid\ \mathbf{0}\end{array}$$

4.3. Bisimulation and Bisimilarity

A highlight of this formalization project is the simplified definitions of bisimilarities using the new coinductive relation package of HOL4. Without this package, bisimilaries can still be defined in HOL, but proving their properties would be more complicated. Below we explain how the weak bisimulation (and bisimilarity) is defined. The way to define strong bisimulation (and bisimilarity) and other concepts (expansion, contraction, etc.) are in the same manner.

To define (weak) bisimilarity, we need to define weak transitions of CCS processes. First of all, a (possibly empty) sequence of τ-transitions between two processes is defined as a new relation `EPS` ($\stackrel{\epsilon}{\Longrightarrow}$), which is the reflexive transitive closure (RTC, denoted by * in HOL4) of ordinary τ-transitions of CCS processes:

[5] If the input process are infinite branching, the function will not terminate.

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

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

$$E\ =u\Rightarrow\ E' \stackrel{\text{def}}{=} \exists\, E_1\ E_2.\ E \stackrel{\epsilon}{\Longrightarrow} E_1 \wedge E_1\ -u\rightarrow\ E_2 \wedge E_2 \stackrel{\epsilon}{\Longrightarrow} E' \qquad \texttt{[WEAK\_TRANS]}$$

The definition of weak bisimulation is based on weak and ϵ–transitions:

$$\begin{array}{l}
\texttt{WEAK_BISIM}\ \mathit{Wbsm} \stackrel{\text{def}}{=} \\
\quad \forall\, E\ E'. \\
\qquad \mathit{Wbsm}\ E\ E' \Longrightarrow \\
\qquad (\forall\, l. \\
\qquad\quad (\forall\, E_1.\ E\ -\texttt{label}\ l\rightarrow\ E_1 \Longrightarrow \exists\, E_2.\ E'\ =\texttt{label}\ l\Rightarrow\ E_2 \wedge \mathit{Wbsm}\ E_1\ E_2) \wedge \\
\qquad\quad \forall\, E_2.\ E'\ -\texttt{label}\ l\rightarrow\ E_2 \Longrightarrow \exists\, E_1.\ E\ =\texttt{label}\ l\Rightarrow\ E_1 \wedge \mathit{Wbsm}\ E_1\ E_2) \wedge \\
\qquad (\forall\, E_1.\ E\ -\tau\rightarrow\ E_1 \Longrightarrow \exists\, E_2.\ E' \stackrel{\epsilon}{\Longrightarrow} E_2 \wedge \mathit{Wbsm}\ E_1\ E_2) \wedge \\
\qquad \forall\, E_2.\ E'\ -\tau\rightarrow\ E_2 \Longrightarrow \exists\, E_1.\ E \stackrel{\epsilon}{\Longrightarrow} E_1 \wedge \mathit{Wbsm}\ E_1\ E_2 \qquad \texttt{[WEAK_BISIM]}
\end{array}$$

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 scripts:

```
CoInductive WEAK_EQUIV :
    !(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'
End
```

Like the case of `TRANS`, a successful invocation of the coinductive definitional principle returns three important theorems (*rules*, *coind* and *cases*):

- *rules* is a conjunction of implications that will be the same as the input term:

$$\begin{array}{l}
\vdash\ \forall\, E\ E'. \\
\qquad (\forall\, l. \\
\qquad\quad (\forall\, E_1.\ E\ -\texttt{label}\ l\rightarrow\ E_1 \Longrightarrow \exists\, E_2.\ E'\ =\texttt{label}\ l\Rightarrow\ E_2 \wedge E_1 \approx E_2) \wedge \\
\qquad\quad \forall\, E_2.\ E'\ -\texttt{label}\ l\rightarrow\ E_2 \Longrightarrow \exists\, E_1.\ E\ =\texttt{label}\ l\Rightarrow\ E_1 \wedge E_1 \approx E_2) \wedge \\
\qquad (\forall\, E_1.\ E\ -\tau\rightarrow\ E_1 \Longrightarrow \exists\, E_2.\ E' \stackrel{\epsilon}{\Longrightarrow} E_2 \wedge E_1 \approx E_2) \wedge \\
\qquad (\forall\, E_2.\ E'\ -\tau\rightarrow\ E_2 \Longrightarrow \exists\, E_1.\ E \stackrel{\epsilon}{\Longrightarrow} E_1 \wedge E_1 \approx E_2) \Longrightarrow \\
\qquad E \approx E' \qquad \texttt{[WEAK_EQUIV_rules]}
\end{array}$$

- *coind* is the coinduction principle for the relation.

$$\begin{array}{l}
\vdash\ \forall\, \mathit{WEAK_EQUIV'}. \\
\qquad (\forall\, a_0\ a_1. \\
\qquad\quad \mathit{WEAK_EQUIV'}\ a_0\ a_1 \Longrightarrow \\
\qquad\quad (\forall\, l. \\
\qquad\qquad (\forall\, E_1. \\
\qquad\qquad\quad a_0\ -\texttt{label}\ l\rightarrow\ E_1 \Longrightarrow \\
\qquad\qquad\quad \exists\, E_2.\ a_1\ =\texttt{label}\ l\Rightarrow\ E_2 \wedge \mathit{WEAK_EQUIV'}\ E_1\ E_2) \wedge \\
\qquad\qquad \forall\, E_2.
\end{array}$$

$$a_1 \xrightarrow{\texttt{label}\ l} E_2 \implies$$
$$\exists E_1.\ a_0 \stackrel{\texttt{label}\ l}{\Longrightarrow} E_1 \wedge \mathit{WEAK\_EQUIV'}\ E_1\ E_2) \wedge$$
$$(\forall E_1.\ a_0 \xrightarrow{\tau} E_1 \implies \exists E_2.\ a_1 \stackrel{\epsilon}{\Longrightarrow} E_2 \wedge \mathit{WEAK\_EQUIV'}\ E_1\ E_2) \wedge$$
$$\forall E_2.\ a_1 \xrightarrow{\tau} E_2 \implies \exists E_1.\ a_0 \stackrel{\epsilon}{\Longrightarrow} E_1 \wedge \mathit{WEAK\_EQUIV'}\ E_1\ E_2) \implies$$
$$\forall a_0\ a_1.\ \mathit{WEAK\_EQUIV'}\ a_0\ a_1 \implies a_0 \approx a_1 \qquad \texttt{[WEAK\_EQUIV\_coind]}$$

- *cases* is the so-called 'cases' or 'inversion' theorem for the relations, and is used to decompose an element in the relation into the possible ways of obtaining it by the rules.

$$\vdash \forall a_0\ a_1.$$
$$a_0 \approx a_1 \iff$$
$$(\forall l.$$
$$(\forall E_1.\ a_0 \xrightarrow{\texttt{label}\ l} E_1 \implies \exists E_2.\ a_1 \stackrel{\texttt{label}\ l}{\Longrightarrow} E_2 \wedge E_1 \approx E_2) \wedge$$
$$\forall E_2.\ a_1 \xrightarrow{\texttt{label}\ l} E_2 \implies \exists E_1.\ a_0 \stackrel{\texttt{label}\ l}{\Longrightarrow} E_1 \wedge E_1 \approx E_2) \wedge$$
$$(\forall E_1.\ a_0 \xrightarrow{\tau} E_1 \implies \exists E_2.\ a_1 \stackrel{\epsilon}{\Longrightarrow} E_2 \wedge E_1 \approx E_2) \wedge$$
$$\forall E_2.\ a_1 \xrightarrow{\tau} E_2 \implies \exists E_1.\ a_0 \stackrel{\epsilon}{\Longrightarrow} E_1 \wedge E_1 \approx E_2 \qquad \texttt{[WEAK\_EQUIV\_cases]}$$

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 ($\approx^c$, `OBS_CONGR`) follows Definition 2.2:

$$E \approx^c E' \stackrel{\text{def}}{=}$$
$$\forall u.$$
$$(\forall E_1.\ E \xrightarrow{u} E_1 \implies \exists E_2.\ E' \stackrel{u}{\Longrightarrow} E_2 \wedge E_1 \approx E_2) \wedge$$
$$\forall E_2.\ E' \xrightarrow{u} E_2 \implies \exists E_1.\ E \stackrel{u}{\Longrightarrow} E_1 \wedge E_1 \approx E_2 \qquad \texttt{[OBS\_CONGR]}$$

Below is the formal version of Lemma 2.3, which is needed in the proof of unique-solution Theorem 3.16:

$$\vdash \texttt{WEAK\_BISIM}\ \mathit{Wbsm} \implies$$
$$\forall E\ E'.$$
$$(\forall u.$$
$$(\forall E_1.\ E \xrightarrow{u} E_1 \implies \exists E_2.\ E' \stackrel{u}{\Longrightarrow} E_2 \wedge \mathit{Wbsm}\ E_1\ E_2) \wedge$$
$$\forall E_2.\ E' \xrightarrow{u} E_2 \implies \exists E_1.\ E \stackrel{u}{\Longrightarrow} E_1 \wedge \mathit{Wbsm}\ E_1\ E_2) \implies$$
$$E \approx^c E' \qquad \texttt{[OBS\_CONGR\_BY\_WEAK\_BISIM]}$$

On the relationship between (weak) bisimilarity and rooted bisimilarity, we have proved Deng's Lemma and Hennessy's Lemma (Lemma 4.1 and 4.2 of [2, p. 176, 178]):

$$\vdash p \approx q \implies (\exists p'.\ p \xrightarrow{\tau} p' \wedge p' \approx q) \vee (\exists q'.\ q \xrightarrow{\tau} q' \wedge p \approx q') \vee p \approx^c q \qquad \texttt{[DENG\_LEMMA]}$$

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

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:

$$\begin{array}{ll}
\texttt{STRONG_SUM_IDEMP:} & \vdash E + E \sim E \\
\texttt{STRONG_SUM_COMM:} & \vdash E + E' \sim E' + E \\
\texttt{STRONG_SUM_IDENT_L:} & \vdash \mathbf{0} + E \sim E \\
\texttt{STRONG_SUM_IDENT_R:} & \vdash E + \mathbf{0} \sim E \\
\texttt{STRONG_SUM_ASSOC_R:} & \vdash E + E' + E'' \sim E + (E' + E'') \\
\texttt{STRONG_SUM_ASSOC_L:} & \vdash E + (E' + E'') \sim E + E' + E'' \\
\texttt{STRONG_SUM_MID_IDEMP:} & \vdash E + E' + E \sim E' + E \\
\texttt{STRONG_LEFT_SUM_MID_IDEMP:} & \vdash E + E' + E'' + E' \sim E + E'' + E'
\end{array}$$

The first five of them are proven by constructing appropriate bisimulation relations, and their formal proofs are written in a goal-directed manner. On the other hand, the last three algebraic 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). These algebraic laws also hold for weak bisimilarity and rooted bisimilarity, as these are coarser than strong bisimilarity. But for weak bisimilarity and rooted bisimilarity, the following so-called τ-laws are further available:

$$\begin{array}{ll}
\texttt{TAU1:} & \vdash u.\tau.E \approx^{c} u.E \\
\texttt{TAU2:} & \vdash E + \tau.E \approx^{c} \tau.E \\
\texttt{TAU3:} & \vdash u.(E + \tau.E') + u.E' \approx^{c} u.(E + \tau.E') \\
\texttt{TAU_STRAT:} & \vdash E + \tau.(E' + E) \approx^{c} \tau.(E' + E) \\
\texttt{TAU_WEAK:} & \vdash \tau.E \approx E
\end{array}$$

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:

$$\begin{array}{l}
\texttt{EXPANSION}\ \mathit{Exp} \stackrel{\text{def}}{=} \\
\quad \forall E\ E'. \\
\qquad \mathit{Exp}\ E\ E' \Longrightarrow \\
\qquad (\forall l. \\
\qquad\quad (\forall E_1.\ E\ -\texttt{label}\ l\rightarrow\ E_1 \Longrightarrow \exists E_2.\ E'\ -\texttt{label}\ l\rightarrow\ E_2 \wedge \mathit{Exp}\ E_1\ E_2) \wedge \\
\qquad\quad \forall E_2.\ E'\ -\texttt{label}\ l\rightarrow\ E_2 \Longrightarrow \exists E_1.\ E\ =\texttt{label}\ l\Rightarrow\ E_1 \wedge \mathit{Exp}\ E_1\ E_2) \wedge \\
\qquad (\forall E_1.\ E\ -\tau\rightarrow\ E_1 \Longrightarrow \mathit{Exp}\ E_1\ E' \vee \exists E_2.\ E'\ -\tau\rightarrow\ E_2 \wedge \mathit{Exp}\ E_1\ E_2) \wedge \\
\qquad \forall E_2.\ E'\ -\tau\rightarrow\ E_2 \Longrightarrow \exists E_1.\ E\ =\tau\Rightarrow\ E_1 \wedge \mathit{Exp}\ E_1\ E_2 \qquad \texttt{[EXPANSION]}
\end{array}$$

$$\vdash P \succeq_{e} Q \iff \exists \mathit{Exp}.\ \mathit{Exp}\ P\ Q \wedge \texttt{EXPANSION}\ \mathit{Exp} \qquad \texttt{[expands\_thm]}$$

$$\begin{array}{l}
\texttt{CONTRACTION}\ \mathit{Con} \stackrel{\text{def}}{=} \\
\quad \forall E\ E'. \\
\qquad \mathit{Con}\ E\ E' \Longrightarrow \\
\qquad (\forall l. \\
\qquad\quad (\forall E_1.\ E\ -\texttt{label}\ l\rightarrow\ E_1 \Longrightarrow \exists E_2.\ E'\ -\texttt{label}\ l\rightarrow\ E_2 \wedge \mathit{Con}\ E_1\ E_2) \wedge \\
\qquad\quad \forall E_2.\ E'\ -\texttt{label}\ l\rightarrow\ E_2 \Longrightarrow \exists E_1.\ E\ =\texttt{label}\ l\Rightarrow\ E_1 \wedge E_1 \approx E_2) \wedge \\
\qquad (\forall E_1.\ E\ -\tau\rightarrow\ E_1 \Longrightarrow \mathit{Con}\ E_1\ E' \vee \exists E_2.\ E'\ -\tau\rightarrow\ E_2 \wedge \mathit{Con}\ E_1\ E_2) \wedge \\
\qquad \forall E_2.\ E'\ -\tau\rightarrow\ E_2 \Longrightarrow \exists E_1.\ E \stackrel{\epsilon}{\Longrightarrow} E_1 \wedge E_1 \approx E_2 \qquad \texttt{[CONTRACTION]}
\end{array}$$

$$\vdash P \succeq_{\text{bis}} Q \iff \exists \mathit{Con}.\ \mathit{Con}\ P\ Q \wedge \texttt{CONTRACTION}\ \mathit{Con} \qquad \texttt{[contracts\_thm]}$$

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

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

1. *(expansion preorder implies contraction preorder)*

   $\vdash\ P\ \succeq_{\mathrm{e}}\ Q\ \Longrightarrow\ P\ \succeq_{\mathrm{bis}}\ Q$ [expands_IMP_contracts]

2. *(contraction preorder implies weak bisimilarity)*

   $\vdash\ P\ \succeq_{\mathrm{bis}}\ Q\ \Longrightarrow\ P\ \approx\ Q$ [contracts_IMP_WEAK_EQUIV]

In general a proof of a property for the contraction (and the contraction preorder) is harder than that for the cases of expansion: this is mostly due to the (surprising) fact that, although the contraction preorder $\succeq_{\mathrm{bis}}$ is contained in bisimilarity ($\approx$), in general it won't be true that if $\mathcal{R}$ is a contraction then $\mathcal{R}$ itself is a bisimulation, i.e. the following proposition does not hold: (what holds is that $\mathcal{R} \cup \approx$ will be a bisimulation)

$\forall\, Con\,.\ \texttt{CONTRACTION}\ Con\ \Longrightarrow\ \texttt{WEAK\_BISIM}\ Con$

For instance, in the proof of `contracts_IMP_WEAK_EQUIV`, we can prove it by constructing a bisimulation *Wbsm* containing two processes P and Q, given that they are in Con (a contraction):

```
    ∃ Wbsm. Wbsm P Q ∧ WEAK_BISIM Wbsm
----------------------------------
 0. Con P Q
 1. CONTRACTION Con
```

We cannot show that Con itself is a bisimulation, but rather that the union of Con and $\approx$ is a bisimulation. The corresponding lemma for the expansion preorder is rather straightforward (just use Con).

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

$$
\begin{array}{l}
E\ \succeq^{\mathrm{c}}_{\mathrm{bis}}\ E' \stackrel{\text{def}}{=} \\
\quad \forall\, u\,. \\
\qquad (\forall\, E_1\,.\ E\ -u\rightarrow\ E_1\ \Longrightarrow\ \exists\, E_2\,.\ E'\ -u\rightarrow\ E_2\ \wedge\ E_1\ \succeq_{\mathrm{bis}}\ E_2)\ \wedge \\
\qquad \forall\, E_2\,.\ E'\ -u\rightarrow\ E_2\ \Longrightarrow\ \exists\, E_1\,.\ E\ =u\Rightarrow\ E_1\ \wedge\ E_1\ \approx\ E_2
\end{array}
\qquad \texttt{[OBS\_contracts]}
$$

4.6. The formalisation of "bisimulation up to bisimilarity"

"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 denote $(P, Q) \in \mathcal{R}$ for any binary relation $\mathcal{R}$. Moreover, $\sim \mathcal{S} \sim$ is the composition of three binary relations: $\sim$, $\mathcal{S}$ and $\sim$. Hence $P \sim \mathcal{S} \sim Q$ means that, there exist P' and Q' such that $P \sim P'$, $P'\ \mathcal{S}\ Q'$ 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 $P \xrightarrow{\mu} P'$ then, for some Q', $Q \xrightarrow{\mu} Q'$ and $P' \sim \mathcal{S} \sim Q'$,*
2. *Whenever $Q \xrightarrow{\mu} Q'$ then, for some P', $P \xrightarrow{\mu} P'$ and $P' \sim \mathcal{S} \sim Q'$.*

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

$\vdash\ \texttt{STRONG\_BISIM\_UPTO}\ Bsm\ \wedge\ Bsm\ P\ Q\ \Longrightarrow\ P\ \sim\ Q$ [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, the *naive* weak bisimulation up to weak bisimilarity is unsound: if one simply replaces all $\sim$ in Def. 4.2 with $\approx$, the resulting "weak bisimulation up" is not contained in $\approx$. [24] There are a few ways to fix this problem, one is the following:

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

1. *Whenever* $P \xrightarrow{\mu} P'$ *then, for some* Q', $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ *and* $P' \sim \mathcal{S} \approx Q'$,
2. *Whenever* $Q \xrightarrow{\mu} Q'$ *then, for some* P', $P \stackrel{\widehat{\mu}}{\Longrightarrow} P'$ *and* $P' \approx \mathcal{S} \sim Q'$.

or formally (for illustrating purposes below),

$$
\begin{array}{l}
\texttt{WEAK\_BISIM\_UPTO}\ \mathit{Wbsm} \stackrel{\text{def}}{=} \\
\quad \forall E\ E'. \\
\qquad \mathit{Wbsm}\ E\ E' \Longrightarrow \\
\qquad (\forall l. \\
\qquad\quad (\forall E_1. \\
\qquad\qquad E\ -\texttt{label}\ l\rightarrow\ E_1 \Longrightarrow \\
\qquad\qquad \exists E_2. \\
\qquad\qquad\quad E'\ =\texttt{label}\ l\Rightarrow\ E_2\ \wedge \\
\qquad\qquad\quad (\texttt{WEAK\_EQUIV} \circ_r \mathit{Wbsm} \circ_r \texttt{STRONG\_EQUIV})\ E_1\ E_2)\ \wedge \\
\qquad\quad \forall E_2. \\
\qquad\qquad E'\ -\texttt{label}\ l\rightarrow\ E_2 \Longrightarrow \\
\qquad\qquad \exists E_1. \\
\qquad\qquad\quad E\ =\texttt{label}\ l\Rightarrow\ E_1\ \wedge \\
\qquad\qquad\quad (\texttt{STRONG\_EQUIV} \circ_r \mathit{Wbsm} \circ_r \texttt{WEAK\_EQUIV})\ E_1\ E_2)\ \wedge \\
\qquad (\forall E_1. \\
\qquad\quad E\ -\tau\rightarrow\ E_1 \Longrightarrow \\
\qquad\quad \exists E_2.\ E' \stackrel{\epsilon}{\Longrightarrow} E_2\ \wedge\ (\texttt{WEAK\_EQUIV} \circ_r \mathit{Wbsm} \circ_r \texttt{STRONG\_EQUIV})\ E_1\ E_2)\ \wedge \\
\qquad \forall E_2. \\
\qquad\quad E'\ -\tau\rightarrow\ E_2 \Longrightarrow \\
\qquad\quad \exists E_1.\ E \stackrel{\epsilon}{\Longrightarrow} E_1\ \wedge\ (\texttt{STRONG\_EQUIV} \circ_r \mathit{Wbsm} \circ_r \texttt{WEAK\_EQUIV})\ E_1\ E_2
\end{array}
$$

Note that the formal represention of $\sim \mathcal{S} \approx$ is "`WEAK_EQUIV` $\circ_r$ *Wbsm* $\circ_r$ `STRONG_EQUIV`" where the order of $\sim$ and $\approx$ seems reverted. This is because, in HOL's notation, the rightmost relation (`STRONG_EQUIV` or $\sim$) in the relational composition is applied first.

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

$\vdash$ `WEAK_BISIM_UPTO` *Bsm* $\wedge$ *Bsm* *P* *Q* $\Longrightarrow$ *P* $\approx$ *Q*

The above version of "bisimulation up to ≈" is not powerful to prove Milner's "unique solution of equations" theorem for ≈ (c.f. [24] for more details). To complete the proof, the following "double-weak" version is actually used:

Definition 4.6. *(Bisimulation up to* $\approx$*, the "double-weak" version)* $\mathcal{S}$ *is a "*bisimulation up to ≈*" if* $P\ \mathcal{S}\ Q$ *implies, for all* μ*,*

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

Theorem 4.7. *If* $\mathcal{S}$ *is a bisimulation up to* $\approx$ *(as by Definition 4.6), then* $\mathcal{S} \subseteq \approx$*:*

$\vdash$ `WEAK_BISIM_UPTO_ALT` *Bsm* $\wedge$ *Bsm* *P* *Q* $\Longrightarrow$ *P* $\approx$ *Q*

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

In this section we give a proof of the second part of Theorem 2.4, i.e. $\approx^{\text{c}}$ is the coarsest congruence contained in ≈. This theorem is of special interests to us, because within our framework it seems impossible to prove it without any limitation. The general form of this theorem can be expressed informally as below [9, 2, 1]:

Proposition 4.8.

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

However, we are only interested in the proposition from right to left, as from left to right it holds trivially by the substitutivity of $\approx^{c}$ of summation and that fact that $\approx^{c}$ implies $\approx$:

$$\vdash \forall p\ q.\ p \approx^{c} q \implies \forall r.\ p + r \approx q + r \qquad \texttt{[COARSEST\_CONGR\_LR]}$$

The standard argument [1] 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, in this project we do not discuss on free and bound names (also called "sorts") of CCS processes. Instead, we assume that all immediate *weak* derivatives of p and q do not use all available action names. We call it the *free action* property:

$$\texttt{free\_action}\ p \stackrel{\text{def}}{=} \exists a.\ \forall p'.\ \neg(p\ =\texttt{label}\ a\Rightarrow\ p') \qquad \texttt{[free\_action\_def]}$$

Below we show how Proposition 4.8 is connected with the statement in Theorem 2.4, and prove it under the free action assumptions.

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

$$\texttt{WEAK\_CONGR} \stackrel{\text{def}}{=} \texttt{CC WEAK\_EQUIV} \qquad \texttt{[WEAK\_CONGR]}$$

$$\texttt{CC}\ R \stackrel{\text{def}}{=} (\lambda g\ h.\ \forall c.\ \texttt{CONTEXT}\ c \implies R\ (c\ g)\ (c\ h)) \qquad \texttt{[CC\_def]}$$

We do not need to put $R\ g\ h$ in the antecedent of CC_def, as this is anyhow obtained from the trivial context $(\lambda x.\ x)$. The next result shows that, for any binary relation R on CCS processes, the composition closure of R is always at least as fine as R: ($\subseteq_r$ stands for *relational subset*.)

$$\vdash \forall R.\ \texttt{CC}\ R \subseteq_r R \qquad \texttt{[CC\_is\_finer]}$$

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

$$\vdash \forall R\ R'.\ \texttt{congruence}\ R' \wedge R' \subseteq_r R \implies R' \subseteq_r \texttt{CC}\ R \qquad \texttt{[CC\_is\_coarsest]}$$

$$\vdash \forall R\ R'.\ \texttt{precongruence}\ R' \wedge R' \subseteq_r R \implies R' \subseteq_r \texttt{CC}\ R \qquad \texttt{[PCC\_is\_coarsest]}$$

Given the central role of the summation, we also consider the relation closure of bisimilarity w.r.t. summation, called *equivalence compatible with summation* (SUM_EQUIV):

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

Rooted bisimilarity $\approx^{c}$ (a congruence contained in $\approx$), is now contained in WEAK_CONGR, which in turn is trivially contained in SUM_EQUIV, as shown in Fig. 6. Thus, to prove (4.8), the crux is to prove that SUM_EQUIV implies rooted bisimilarity ($\approx^{c}$), making all three relations ($\approx^{c}$, WEAK_CONGR and SUM_EQUIV) coincide:

$$\forall p\ q.\ (\forall r.\ p + r \approx q + r) \implies p \approx^{c} q\ . \tag{2}$$

Under the free action assumptions, the actual formalisation of (2) says:

Theorem 4.9 (COARSEST_CONGR_RL)**.** *Under the free actions assumption, $\approx^{c}$ is coarsest congruence contained in $\approx$.*

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

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

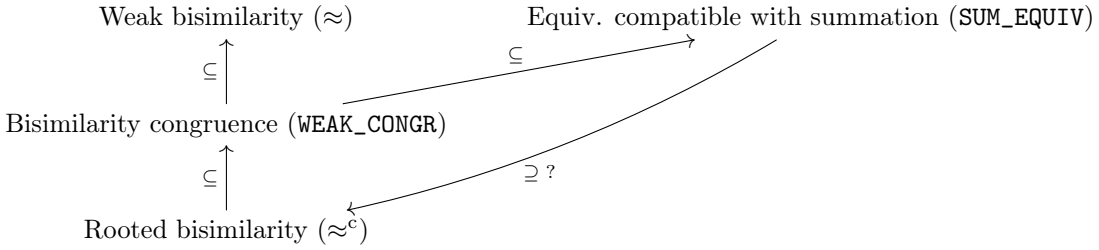

Figure 6: Relationships between several equivalences and ≈

Theorem 4.10 (COARSEST_PRECONGR_RL)**.** *Under the free actions assumption,* $\succeq_{\text{bis}}$ *is the coarsest precongruence contained in* $\succeq_{\text{bis}}$*.*

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

The formal proofs of the above two results follow Milner's proof [1]. The assumption in Milner's result requires $\text{fn}(p) \cup \text{fn}(q) \neq \mathscr{L}$ (where fn stands for *free names*), and is therefore strictly stronger than the assumption in our result above. Indeed, in our result we only look at immediate weak derivatives for p and q, and only requires that there is an input or output action that never appears as a label. Furthermore, such an action need not to be the same for p and for q, as an action a and its complementary $\overline{a}$ are considered as *different* actions.

4.8. Arbitrarily many non-bisimilar processes

In this section, we give another more complex proof of the second part of Theorem 2.4, but only for finite-state CCS:

Theorem 4.11 (COARSEST_CONGR_FINITE)**.** *For finite-state CCS,* $\approx$ *is the coarsest congruence contained in* $\approx$*:*

$$\vdash \forall p\ q.\ \texttt{finite\_state}\ p \wedge \texttt{finite\_state}\ q \Longrightarrow (p \approx^{\text{c}} q \iff \forall r.\ p + r \approx q + r)$$

The proof of above theorem was given by Rob van Glabbeek [9], which originally applies to all CCS processes. 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 ($\approx^{\text{c}}$): (c.f. [25] for more details of the proof)

$$\begin{array}{l}\vdash (\exists k.\\ \qquad \texttt{STABLE}\ k \wedge (\forall p'\ u.\ p =u\Rightarrow p' \Longrightarrow \neg(p' \approx k)) \wedge\\ \qquad \forall q'\ u.\ q =u\Rightarrow q' \Longrightarrow \neg(q' \approx k)) \Longrightarrow\\ \quad (\forall r.\ p + r \approx q + r) \Longrightarrow\\ \quad p \approx^{\text{c}} q \end{array} \qquad \texttt{[PROP3\_COMMON]}$$

$$\texttt{STABLE}\ p \stackrel{\text{def}}{=} \forall u\ p'.\ p \xrightarrow{u} p' \Longrightarrow u \neq \tau \qquad \texttt{[STABLE]}$$

To actually 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 [9]) that are all pairwise non-bisimilar. We have only partially formalised this proof (Theorem 4.11 in next section), because "the typed logic implemented in various HOL systems (including Isabelle/HOL) is not strong enough to define a type for all possible ordinals" [26]. As a consequence, the final result is only for finite-state CCS — a restricted class.

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.

There could be infinitely many 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 [9]. We call "Klop processes" the processes in this construction:

Definition 4.12 (Klop processes)**.** *For each ordinal λ, and an arbitrary chosen action $a \neq \tau$, 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 4.13. *(Klop processes as recursive function on natural numbers)*

$$\texttt{KLOP}\ a\ 0 \stackrel{\text{def}}{=} \mathbf{0}$$
$$\texttt{KLOP}\ a\ (\texttt{SUC}\ n) \stackrel{\text{def}}{=} \texttt{KLOP}\ a\ n\ +\ \texttt{label}\ a.\texttt{KLOP}\ a\ n \qquad \texttt{[KLOP\_def]}$$

Following the inductive structure of the definition of Klop processes, and using the SOS rules (Sum_1) and (Sum_2), we proved the following properties of Klop functions:

Proposition 4.14. *(Properties of Klop functions and processes)*

1. *(All Klop processes are stable)*

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

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

   $\vdash \texttt{KLOP}\ a\ n\ -\texttt{label}\ a\rightarrow\ E \iff \exists m.\ m < n \wedge E = \texttt{KLOP}\ a\ m$ [KLOP_PROP1]

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

   $\vdash \texttt{KLOP}\ a\ n\ =\texttt{label}\ a\Rightarrow\ E \iff \exists m.\ m < n \wedge E = \texttt{KLOP}\ a\ m$ [KLOP_PROP1']

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

   $\vdash m < n \implies \neg(\texttt{KLOP}\ a\ m \sim \texttt{KLOP}\ a\ n)$ [KLOP_PROP2]

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

   $\vdash m < n \implies \neg(\texttt{KLOP}\ a\ m \approx \texttt{KLOP}\ a\ n)$ [KLOP_PROP2']

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

   $\vdash \texttt{ONE\_ONE}\ (\texttt{KLOP}\ a)$ [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 4.15. *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 $R \Longrightarrow$
FINITE $A \wedge$ INFINITE $B \wedge (\forall x\ y.\ x \in B \wedge y \in B \wedge x \neq y \Longrightarrow \neg R\ x\ y) \Longrightarrow$
$\exists k.\ k \in B \wedge \forall n.\ n \in A \Longrightarrow \neg R\ n\ k$ [INFINITE_EXISTS_LEMMA]

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

Definition 4.16. *(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 $\stackrel{\text{def}}{=} (\lambda E\ E'.\ \exists u.\ E\ -u\rightarrow\ E')^*$ [Reach_def]

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

   NODES $p \stackrel{\text{def}}{=} \{q \mid \text{Reach}\ p\ q\}$ [NODES_def]

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

   finite_state $p \stackrel{\text{def}}{=}$ FINITE (NODES p) [finite_state_def]

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

Proposition 4.17. *If p has a weak transition onto q, then q must be in the node set of p:*

$\vdash\ p\ =u\Rightarrow\ q \Longrightarrow q \in$ NODES p [WEAK_TRANS_IN_NODES]

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

Lemma 4.18. *(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):*

$\vdash \forall p\ q.$
finite_state $p \wedge$ finite_state $q \Longrightarrow$
$\exists k.$
STABLE $k \wedge (\forall p'\ u.\ p\ =u\Rightarrow\ p' \Longrightarrow \neg(p' \approx k)) \wedge$
$\forall q'\ u.\ q\ =u\Rightarrow\ q' \Longrightarrow \neg(q' \approx k)$ [KLOP_LEMMA_FINITE]

Combining the above lemma with PROP3_COMMON and COARSEST_CONGR_RL, we can prove the theorem COARSEST_CONGR_FINITE.

4.9. Context, guardedness and (pre)congruence

We have chosen to use λ-expressions (with the type "(α, β) CCS $\rightarrow$ (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:

$\vdash$ CONTEXT $(\lambda t.\ t) \wedge (\forall p.$ CONTEXT $(\lambda t.\ p)) \wedge$
$(\forall a\ e.$ CONTEXT $e \Longrightarrow$ CONTEXT $(\lambda t.\ a.e\ t)) \wedge$
$(\forall e_1\ e_2.$ CONTEXT $e_1 \wedge$ CONTEXT $e_2 \Longrightarrow$ CONTEXT $(\lambda t.\ e_1\ t + e_2\ t)) \wedge$
$(\forall e_1\ e_2.$ CONTEXT $e_1 \wedge$ CONTEXT $e_2 \Longrightarrow$ CONTEXT $(\lambda t.\ e_1\ t \mid e_2\ t)) \wedge$
$(\forall L\ e.$ CONTEXT $e \Longrightarrow$ CONTEXT $(\lambda t.\ (\nu L)\ (e\ t))) \wedge$
$\forall rf\ e.$ CONTEXT $e \Longrightarrow$ CONTEXT $(\lambda t.$ relab $(e\ t)\ rf)$ [CONTEXT_rules]

Notice in above inductive definitions, for any CCS term p, $\lambda\, t.\ p$ is a valid context. In this case there is no hole in the context. On the other hand, if any CCS term with recursion could be a context, the recursion must be a subterm of this p.

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

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

(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 *guarded* (`SG`) if each hole is underneath a *visible* prefix:

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

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 [1] for the informal definitions.)

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

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):

$$\begin{array}{l}\vdash \texttt{GCONTEXT}\ (\lambda\, t.\ t) \wedge (\forall\, p.\ \texttt{GCONTEXT}\ (\lambda\, t.\ p)) \wedge \\ \quad (\forall\, a\ e.\ \texttt{GCONTEXT}\ e \Longrightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ a.e\ t)) \wedge \\ \quad (\forall\, a_1\ a_2\ e_1\ e_2.\ \\ \qquad \texttt{GCONTEXT}\ e_1 \wedge \texttt{GCONTEXT}\ e_2 \Longrightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ a_1.e_1\ t + a_2.e_2\ t)) \wedge \\ \quad (\forall\, e_1\ e_2.\ \texttt{GCONTEXT}\ e_1 \wedge \texttt{GCONTEXT}\ e_2 \Longrightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ e_1\ t \mid e_2\ t)) \wedge \\ \quad (\forall\, L\ e.\ \texttt{GCONTEXT}\ e \Longrightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ (\nu L)\ (e\ t))) \wedge \\ \quad \forall\, rf\ e.\ \texttt{GCONTEXT}\ e \Longrightarrow \texttt{GCONTEXT}\ (\lambda\, t.\ \texttt{relab}\ (e\ t)\ rf) \qquad \texttt{[GCONTEXT\_rules]}\end{array}$$

$$\begin{array}{l}\vdash (\forall\, p.\ \texttt{WGS}\ (\lambda\, t.\ p)) \wedge (\forall\, a\ e.\ \texttt{GCONTEXT}\ e \Longrightarrow \texttt{WGS}\ (\lambda\, t.\ a.e\ t)) \wedge \\ \quad (\forall\, a_1\ a_2\ e_1\ e_2.\ \texttt{GCONTEXT}\ e_1 \wedge \texttt{GCONTEXT}\ e_2 \Longrightarrow \texttt{WGS}\ (\lambda\, t.\ a_1.e_1\ t + a_2.e_2\ t)) \wedge \\ \quad (\forall\, e_1\ e_2.\ \texttt{WGS}\ e_1 \wedge \texttt{WGS}\ e_2 \Longrightarrow \texttt{WGS}\ (\lambda\, t.\ e_1\ t \mid e_2\ t)) \wedge \\ \quad (\forall\, L\ e.\ \texttt{WGS}\ e \Longrightarrow \texttt{WGS}\ (\lambda\, t.\ (\nu L)\ (e\ t))) \wedge \\ \quad \forall\, rf\ e.\ \texttt{WGS}\ e \Longrightarrow \texttt{WGS}\ (\lambda\, t.\ \texttt{relab}\ (e\ t)\ rf) \qquad \texttt{[WGS\_rules]}\end{array}$$

Many lemmas about the above concepts (`CONTEXT`, `WG`, `SEQ`, etc.) have to be proved for the relationships among these kinds of contexts and properties about their compositions. These proofs are usually tedious and 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):

$$\begin{array}{l}\texttt{congruence}\ R \stackrel{\text{def}}{=} \\ \quad \texttt{equivalence}\ R \wedge \forall\, x\ y\ ctx.\ \texttt{CONTEXT}\ ctx \Longrightarrow R\ x\ y \Longrightarrow R\ (ctx\ x)\ (ctx\ y) \qquad \texttt{[congruence]}\end{array}$$

800 `precongruence` $R \stackrel{\text{def}}{=}$
`PreOrder` $R \wedge \forall x\ y\ ctx.$ `CONTEXT` $ctx \Longrightarrow R\ x\ y \Longrightarrow R\ (ctx\ x)\ (ctx\ y)$ [precongruence]

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

$\vdash$ `congruence STRONG_EQUIV` [STRONG_EQUIV_congruence]
$\vdash$ `congruence OBS_CONGR` [OBS_CONGR_congruence]

805 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 guarded sums.

Rooted contraction, on the other hand, is a precongruence:

$\vdash$ `precongruence OBS_contracts` [OBS_contracts_precongruence]

4.10. Unique solution of equations

810 *4.10.1. Strong bisimilarity*

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 4.19 (`STRONG_UNIQUE_SOLUTION_LEMMA` (Lemma 3.13 of [1], univariable version)). *If the variable X is weakly guarded in E, and $E\{P/X\} \xrightarrow{\alpha} P'$, then P' takes the form $E'\{P/X\}$ (for some expression E'),* 815 *and moreover, for any Q, $E\{Q/X\} \xrightarrow{\alpha} E'\{Q/X\}$:*

$\vdash$ `WG` $E \Longrightarrow$
$\forall P\ a\ P'.\ E\ P\ -a\rightarrow\ P' \Longrightarrow \exists E'.$ `CONTEXT` $E' \wedge P' = E'\ P \wedge \forall Q.\ E\ Q\ -a\rightarrow\ E'\ Q$

Then, by a structural induction on weakly-guarded context (`WG`), we have proved Milner's strong unique-solution theorem in the univariable case (c.f. Section 5 for the multivariable case). The formal proof basically 820 follows the outline of its informal version. It is tedious due to large amounts of case analyses on each CCS operator.

Theorem 4.20 (`STRONG_UNIQUE_SOLUTION` (Proposition 3.14 of [1])). *Suppose the expression E contains at most the variable X, and let X be weakly guarded in E.*

$$\text{If } P \sim E\{P/X\} \text{ and } Q \sim E\{Q/X\} \text{ then } P \sim Q. \tag{3}$$

$\vdash$ `WG` $E \wedge P \sim E\ P \wedge Q \sim E\ Q \Longrightarrow P \sim Q$

4.10.2. Weak bisimilarity

Milner's book [1] contains only two "unique solutions of equations" theorems, one for strong equivalence, 825 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 a congruence, we have to be more restrictive on the syntax for the equation, using strong guardedness.

Lemma 4.21 (`WEAK_UNIQUE_SOLUTION_LEMMA`). *If the variable X is guarded and sequential in G, and $G\{P/X\} \xrightarrow{\alpha} P'$, then P' takes the form $H\{P/X\}$, for some expression H, and for any Q, we have* 830 *$G\{Q/X\} \xrightarrow{\alpha} H\{Q/X\}$. Moreover H is sequential, and if $\alpha = \tau$ then H is also guarded.*

$\vdash$ `SG` $G \wedge$ `GSEQ` $G \Longrightarrow$
$\forall P\ a\ P'.$
$G\ P\ -a\rightarrow\ P' \Longrightarrow$
$\exists H.$ `GSEQ` $H \wedge (a = \tau \Longrightarrow$ `SG` $H) \wedge P' = H\ P \wedge \forall Q.\ G\ Q\ -a\rightarrow\ H\ Q$

835 We can then derive the target theorem:

Theorem 4.22 (`WEAK_UNIQUE_SOLUTION`). *Let E be guarded and sequential expressions, and let $P \approx E\{P/X\}$, $Q \approx E\{Q/X\}$. Then $P \approx Q$.*

$\vdash$ `SG` $E \wedge$ `GSEQ` $E \wedge P \approx E\ P \wedge Q \approx E\ Q \Longrightarrow P \approx Q$

4.10.3. 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 4.23 (`OBS_CONGR_BY_WEAK_BISIM`). $\vdash$ `WEAK_BISIM` $Wbsm \implies$
$\forall E\ E'.$
$(\forall u.$
$(\forall E_1.\ E\ -u\rightarrow\ E_1 \implies \exists E_2.\ E'\ =u\Rightarrow\ E_2 \wedge\ Wbsm\ E_1\ E_2) \wedge$
$\forall E_2.\ E'\ -u\rightarrow\ E_2 \implies \exists E_1.\ E\ =u\Rightarrow\ E_1 \wedge\ Wbsm\ E_1\ E_2) \implies$
$E \approx^{c} E'$

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 4.24 (`OBS_UNIQUE_SOLUTION_LEMMA`). *If the variable X is guarded and sequential in G, and $G\{P/X\} \xrightarrow{\alpha} P'$, then P' takes the form $H\{P/X\}$, for some H, and for any Q, we also have $G\{Q/X\} \xrightarrow{\alpha} H\{Q/X\}$. Moreover H is sequential, and if $\alpha = \tau$ then H is also guarded.*

$\vdash$ `SG` $G \wedge$ `SEQ` $G \implies$
$\forall P\ a\ P'.$
$G\ P\ -a\rightarrow\ P' \implies$
$\exists H.$ `SEQ` $H \wedge (a = \tau \implies$ `SG` $H) \wedge P' = H\ P \wedge \forall Q.\ G\ Q\ -a\rightarrow\ H\ Q$

Theorem 4.25 (`OBS_UNIQUE_SOLUTION`). *Let E be guarded and sequential expressions, and let $P \approx^{c} E\{P/X\}$, $Q \approx^{c} E\{Q/X\}$. Then $P \approx^{c} Q$.*

$\vdash$ `SG` $E \wedge$ `SEQ` $E \wedge P \approx^{c} E\ P \wedge Q \approx^{c} E\ Q \implies P \approx^{c} Q$

4.11. Unique solution of contractions

A delicate point in the formalisation of the results about unique solution of contractions are the proof of Lemma 3.14 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` $R\ a\ l\ b \stackrel{\text{def}}{=}$
$\forall P.$
$(\forall x.\ P\ x\ []\ x) \wedge$
$(\forall x\ h\ y\ t\ z.\ R\ x\ h\ y \wedge P\ y\ t\ z \implies P\ x\ (h::t)\ z) \implies$
$P\ a\ l\ b$ `[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` $\stackrel{\text{def}}{=}$ `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):

$$\texttt{NO\_LABEL}\ L \stackrel{\textsf{def}}{=} \neg\exists l.\ \texttt{MEM}\ (\texttt{label}\ l)\ L \qquad \texttt{[NO\_LABEL\_def]}$$

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:

$$\begin{array}{l}\texttt{UNIQUE\_LABEL}\ u\ L \stackrel{\textsf{def}}{=} \\ \quad \exists L_1\ L_2.\ L_1 +\!\!+ [u] +\!\!+ L_2 = L \wedge \texttt{NO\_LABEL}\ L_1 \wedge \texttt{NO\_LABEL}\ L_2 \qquad \texttt{[UNIQUE\_LABEL\_def]}\end{array}$$

The final relationship between traces and weak transitions is stated and proved in the following theorem (where the variable *acts* stands for a list of actions); it says, a weak transition $P \stackrel{u}{\Rightarrow} P'$ is also a trace $P \stackrel{acts}{\longrightarrow} P'$ with a non-empty action list *acts*, in which either there is no label (for $u = \tau$), or u is the unique label (for $u \neq \tau$):

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

Now the formal version of Lemma 3.10 (`UNIQUE_SOLUTION_OF_CONTRACTIONS_LEMMA`):

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

Traces are actually used in the proof of above lemma via the following lemma (`unfolding_lemma4`):

$$\begin{array}{l}\vdash \texttt{GCONTEXT}\ C \wedge \texttt{WGS}\ E \wedge \texttt{TRACE}\ ((C \circ \texttt{FUNPOW}\ E\ n)\ P)\ xs\ P' \wedge \texttt{LENGTH}\ xs \leq n \Longrightarrow \\ \exists C'.\ \texttt{GCONTEXT}\ C' \wedge P' = C'\ P \wedge \forall Q.\ \texttt{TRACE}\ ((C \circ \texttt{FUNPOW}\ E\ n)\ Q)\ xs\ (C'\ Q)\end{array}$$

which roughly says, for any context C and weakly-guarded context E, if $C[\,E^n[P]\,] \stackrel{xs}{\Longrightarrow} P'$ and the length of actions $xs \leqslant n$, then P' has the form of $C'[P]$. Traces are used in reasoning about the number of intermediate actions in weak transitions. For instance, from Def. 3.7, it is easy to see that, a weak transition either becomes shorter or remains the same when moving between $\succeq_{\text{bis}}$-related processes. This property is essential in the proof of Lemma 3.10. We show only one such lemma, for the case of non-τ weak transitions passing into $\succeq_{\text{bis}}$ (from left to right):

$$\begin{array}{l}\vdash P \succeq_{\text{bis}} Q \Longrightarrow \\ \quad \forall xs\ l\ P'. \\ \qquad \texttt{TRACE}\ P\ xs\ P' \wedge \texttt{UNIQUE\_LABEL}\ (\texttt{label}\ l)\ xs \Longrightarrow \\ \qquad \exists xs'\ Q'. \\ \qquad\quad \texttt{TRACE}\ Q\ xs'\ Q' \wedge P \succeq_{\text{bis}} Q \wedge \texttt{LENGTH}\ xs' \leq \texttt{LENGTH}\ xs \wedge \\ \qquad\quad \texttt{UNIQUE\_LABEL}\ (\texttt{label}\ l)\ xs' \qquad \texttt{[contracts\_AND\_TRACE\_label]}\end{array}$$

With all above lemmas, we can thus finally prove Theorem 3.11:

$$\vdash \texttt{WGS}\ E \wedge P \succeq_{\text{bis}} E\ P \wedge Q \succeq_{\text{bis}} E\ Q \Longrightarrow P \approx Q \qquad \texttt{[UNIQUE\_SOLUTION\_OF\_CONTRACTIONS]}$$

4.12. Unique solution of rooted contractions

The formal proof of "unique solution of rooted contractions theorem" (Theorem 3.16) has the same initial proof steps as Theorem 3.11; 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 3.16 (having proved the precongruence of rooted contraction, we can use weakly-guarded expressions, without constraints on sums; that is, `WG` in place of `WGS`):

$$\vdash \texttt{WG}\ E\ \wedge\ P\ \succeq^{\mathrm{c}}_{\mathrm{bis}}\ E\ P\ \wedge\ Q\ \succeq^{\mathrm{c}}_{\mathrm{bis}}\ E\ Q \implies P \approx^{\mathrm{c}} Q\ \texttt{[UNIQUE\_SOLUTION\_OF\_ROOTED\_CONTRACTIONS]}$$

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 guarded and sequential.

5. The multivariable formalisation

In this section, we discuss the formalisation of the multivariable cases of Milner's "unique solution of equations (for $\sim$)" (Theorem 3.4) and our "unique solution of rooted contractions" (Theorem 3.16). This formalisation only supports finitely many equations/contractions and equation variables.[6] We chose these two theorems because they both depend on the same concept of "weakly guarded contexts", and Lemma 3.15. The work is an extension of the univariable case [3] in the sense that neither the CCS datatype nor the SOS rules need to be changed.

The central problem of the multivariable formalisation is the representation of CCS equations (expressions, contexts). In the univariable formalisation, λ-functions (of type $CCS \to CCS$) are used to represent univariable CCS equations, and variable substitutions are simply calls to λ-functions on the target CCS terms. This idea cannot be extended to the multivariable case, as we do not have a fixed number of variables to deal with.

In the literature the variables X_i of a system of equations $\{X_i = E_i\}_{i\in I}$ are usually considered as *equation variables outside* of the CCS syntax; then an equation body E_i is an *open* expression built with CCS operators plus the equation variables. For a formal point of view, this means we either need to define a whole new datatype (with extra equation variables) in which each CCS opeartor must be duplicated, or we have to add equation variables as a new primitive (e.g. `Var X`) to the existing CCS type (besides the agent variables such as `var X`). Then the substitution of equation variables in $E_i\{\widetilde{P}/\widetilde{X}\}$ would be the *syntactic substitution* of each occurrence of X_i in E_i with the corresponding P_i. In either cases the *disjointness* between equation variables and agent variables is syntactically guaranteed. Both solutions are rather combersome, and require a non-trivial modification of the formalisation of the univariable case.

In this project, we have followed Milner [1], using the same alphabet for both agent variables and equation variables. This has allowed a large reuse of the univariable formalisation. For instance, the variable substitution function used in SOS rules [`REC`] can be also used for substituting eqaution variables. However, a lot of care is needed in the proofs of many fundamental lemmas, mostly due to the variable capture issues between equation variables and agent variables.

5.1. Free and bound variables

Recall the beginning of Section 3, the CCS syntax leaves the syntactic possibility that an agent variable A occurs outside the recursion operator of the same variable. In Milner's original book chapter [10], the variables which occur in Recursion (such as X in $\texttt{rec}\, X.\, E$) are called *bound* variables, while those which occur unbound are called *free* variables.

[6] The original theorems hold for even (uncountably) infinite equations and equation variables.

We denote the set of all bound variables of a given CCS term E (we recall that these are variables that are bound in a recursion subexpression of E) as $\mathsf{bv}(E)$ (or $\texttt{BV}\ E$ in HOL), and the set of free variables as $\mathsf{fv}(E)$ (or $\texttt{FV}\ E$ in HOL). Both `BV` and `FV` have the type "$(\alpha,\ \beta)\ \texttt{CCS} \to \alpha \to \texttt{bool}$", i.e. functions taking CCS terms returning sets of variables (of type α). For their definition the interesting cases are those of recursion and agent variables, shown below (here `DELETE` and `INSERT` are set-theoretic operators of HOL's `pred_set` theory):

$\texttt{FV}\ (\texttt{var}\ X) \stackrel{\text{def}}{=} \{X\}$	$\texttt{FV}\ (\texttt{rec}\ X\ p) \stackrel{\text{def}}{=} \texttt{FV}\ p\ \texttt{DELETE}\ X$
$\texttt{BV}\ (\texttt{var}\ X) \stackrel{\text{def}}{=} \emptyset$	$\texttt{BV}\ (\texttt{rec}\ X\ p) \stackrel{\text{def}}{=} X\ \texttt{INSERT}\ \texttt{BV}\ p$

Furthermore, E is a process, written $\texttt{IS\_PROC}\ E$, if it does not contain any free variable, i.e. $\mathsf{fv}(E) = \emptyset$:

$$\texttt{IS\_PROC}\ E \stackrel{\text{def}}{=} \texttt{FV}\ E = \emptyset \qquad \texttt{[IS\_PROC\_def]}$$

And a list of CCS processes can be asserted by `ALL_PROC` defined upon `IS_PROC`:

$$\texttt{ALL\_PROC}\ Es \stackrel{\text{def}}{=} \texttt{EVERY}\ \texttt{IS\_PROC}\ Es \qquad \texttt{[ALL\_PROC\_def]}$$

Note that, in CCS expression, the set of free and bound variables need not be disjoint, e.g. in $X + \texttt{rec}\ X.E$. More importantly, when going from a CCS expression to its sub-expressions, the set of free variables may increase, while the set of bound variables either remains the same or decreases. For instance, $\mathsf{fv}(\texttt{rec}\ X.(\mu.X)) = \emptyset$, while $\mathsf{fv}(\mu.X) = \{X\}$. This property of $\mathsf{fv}(\cdot)$ brings some difficulities when doing proofs about CCS transitions. As an evidence, we prove the following fundamental result about $\mathsf{fv}(\cdot)$:

Proposition 5.1. *The derivatives of a process are themselves processes, i.e. if* $E \xrightarrow{\mu} E'$ *and* $\mathsf{fv}(E) = \emptyset$, *then* $\mathsf{fv}(E') = \emptyset$. *Or, formally:*

$$\vdash \forall E\ u\ E'.\ E\ -u\to\ E' \wedge \texttt{IS\_PROC}\ E \Longrightarrow \texttt{IS\_PROC}\ E' \qquad \texttt{[TRANS\_PROC]}$$

Proof. We prove a stronger result, i.e., the set of free variables decreases through derivatives:

$$\vdash \forall E\ u\ E'.\ E\ -u\to\ E' \Longrightarrow \texttt{FV}\ E' \subseteq \texttt{FV}\ E \qquad \texttt{[TRANS\_FV]}$$

Milner said it can be proved by "an easy action induction" [10]. From HOL's viewpoint, the so-called "action induction" is actually a higher-order application of the following *induction principle* generated together with SOS rules (which essentially says that `TRANS` is the smallest relation satisfying SOS rules):

$$\begin{aligned}
&\vdash \forall P.\\
&\quad (\forall E\ u.\ P\ (u.E)\ u\ E) \wedge (\forall E\ u\ E_1\ E'.\ P\ E\ u\ E_1 \Longrightarrow P\ (E + E')\ u\ E_1) \wedge\\
&\quad (\forall E\ u\ E_1\ E'.\ P\ E\ u\ E_1 \Longrightarrow P\ (E' + E)\ u\ E_1) \wedge\\
&\quad (\forall E\ u\ E_1\ E'.\ P\ E\ u\ E_1 \Longrightarrow P\ (E \mid E')\ u\ (E_1 \mid E')) \wedge\\
&\quad (\forall E\ u\ E_1\ E'.\ P\ E\ u\ E_1 \Longrightarrow P\ (E' \mid E)\ u\ (E' \mid E_1)) \wedge\\
&\quad (\forall E\ l\ E_1\ E'\ E_2.\\
&\qquad P\ E\ (\texttt{label}\ l)\ E_1 \wedge P\ E'\ (\texttt{label}\ (\texttt{COMPL}\ l))\ E_2 \Longrightarrow\\
&\qquad P\ (E \mid E')\ \tau\ (E_1 \mid E_2)) \wedge\\
&\quad (\forall E\ u\ E'\ l\ L.\\
&\qquad P\ E\ u\ E' \wedge (u = \tau \vee u = \texttt{label}\ l \wedge l \notin L \wedge \texttt{COMPL}\ l \notin L) \Longrightarrow\\
&\qquad P\ ((\nu L)\ E)\ u\ ((\nu L)\ E')) \wedge\\
&\quad (\forall E\ u\ E'\ rf.\ P\ E\ u\ E' \Longrightarrow P\ (\texttt{relab}\ E\ rf)\ (\texttt{relabel}\ rf\ u)\ (\texttt{relab}\ E'\ rf)) \wedge\\
&\quad (\forall E\ u\ X\ E_1.\ P\ ([\texttt{rec}\ X\ E/X]\ E)\ u\ E_1 \Longrightarrow P\ (\texttt{rec}\ X\ E)\ u\ E_1) \Longrightarrow\\
&\quad \forall a_0\ a_1\ a_2.\ a_0\ -a_1\to\ a_2 \Longrightarrow P\ a_0\ a_1\ a_2 \qquad \texttt{[TRANS_IND]}
\end{aligned}$$

The above long theorem has the form $\forall P.\ X \Longrightarrow \forall E\ u\ E'.\ E\ -u\to\ E' \Longrightarrow P\ E\ u\ E'$ (with a_0, a_1, a_2 renamed), where the outermost universal quantifier P is a higher-order proposition (about E, μ and E'), and X is another higher-order proposition about P. What we are trying to prove is actually in the form of $\forall E\ u\ E'.\ E\ -u\to\ E' \Longrightarrow P\ E\ u\ E'$, where $P = (\lambda E\ u\ E'.\ \texttt{FV}\ E' \subseteq \texttt{FV}\ E)$. Thus, if we can prove X under

this specific P, by Modus Ponens (MP) the original proof would have completed. This is the essence of action induction. Now the original goal can be reduced to several conjunct subgoals, each corresponding to one SOS rule. For instance, in the subgoal for `SUM1` we need to prove $\mathsf{fv}(E_1) \subseteq \mathsf{fv}(E) \Longrightarrow \mathsf{fv}(E_1) \subseteq \mathsf{fv}(E+E')$, which holds as $\mathsf{fv}(E) \subseteq \mathsf{fv}(E + E') = \mathsf{fv}(E) \cup \mathsf{fv}(E')$. Eventually we have only the following goal left: (the term above the dash line is the goal, those below the line are assumptions)

```
    FV E' ⊆ FV E DELETE X
------------------------------------
 0. FV E' ⊆ FV ([rec X E/X] E)
```

Notice that, $\texttt{FV}\ E\ \texttt{DELETE}\ X = \texttt{FV}\ (\texttt{rec}\ X\ E)$, so this is indeed the consequence of action induction on the Recursion. Here the problem is that we know nothing about $\texttt{FV}\ ([\texttt{rec}\ X\ E/X]\ E)$. To further proceed, we need to prove some other (easier) lemmas. First, the next lemma can be proven by induction on E and a few basic set-theoretic facts:

$$\vdash \forall X\ E\ E'.\ \texttt{FV}\ ([E'/X]\ E) \subseteq \texttt{FV}\ E \cup \texttt{FV}\ E' \qquad \texttt{[FV\_SUBSET]}$$

Now, if we take $E' = \texttt{rec}\ X.\ E$ in the above lemma, we get $\texttt{FV}\ ([\texttt{rec}\ X\ E/X]\ E) \subseteq \texttt{FV}\ E \cup \texttt{FV}\ (\texttt{rec}\ X\ E) = \texttt{FV}\ E \cup (\texttt{FV}\ E\ \texttt{DELETE}\ X) = \texttt{FV}\ E$, i.e. the following lemma:

$$\vdash \forall X\ E.\ \texttt{FV}\ ([\texttt{rec}\ X\ E/X]\ E) \subseteq \texttt{FV}\ E \qquad \texttt{[FV\_SUBSET\_REC]}$$

Thus we can enrich the assumptions of the current proof goal with above lemma, and obtain $\texttt{FV}\ E' \subseteq \texttt{FV}\ E$ by the transitivity of $\subseteq$:

```
    FV E' ⊆ FV E DELETE X
------------------------------------
 0. FV E' ⊆ FV ([rec X E/X] E)
 1. FV ([rec X E/X] E) ⊆ FV E
 2. FV E' ⊆ FV E
```

Knowing $\texttt{FV}\ E' \subseteq \texttt{FV}\ E$ we cannot prove $\texttt{FV}\ E' \subseteq \texttt{FV}\ E\ \texttt{DELETE}\ X$. However, if we knew $X \notin \texttt{FV}\ E'$, then $\texttt{FV}\ E'\ \texttt{DELETE}\ X = \texttt{FV}\ E'$, and then $\texttt{FV}\ E' \subseteq \texttt{FV}\ E \Longrightarrow \texttt{FV}\ E'\ \texttt{DELETE}\ X \subseteq \texttt{FV}\ E\ \texttt{DELETE}\ X$, no matter if $X \in \texttt{FV}\ E$ or not, and the proof would have completed. Thus it remains to show that

```
    X ∉ FV E'
------------------------------------
 0. FV E' ⊆ FV ([rec X E/X] E)
 1. FV ([rec X E/X] E) ⊆ FV E
 2. FV E' ⊆ FV E
```

Now we try the proof by contradiction (*reductio ad absurdum*): if the goal does not hold, i.e. $X \in \texttt{FV}\ E'$, then by assumption 0 we have $X \in \texttt{FV}\ ([\texttt{rec}\ X\ E/X]\ E)$:

```
    F
------------------------------------
 0. FV E' ⊆ FV ([rec X E/X] E)
 1. FV ([rec X E/X] E) ⊆ FV E
 2. FV E' ⊆ FV E
 3. X ∈ FV E'
 4. X ∈ FV ([rec X E/X] E)
```

But this is impossible, because all free occurrences of X in E now become bound in the form of $\texttt{rec}\ X.\ E$. In fact, the following lemma can be proven by induction on E:

$$\vdash \forall X\ E\ E'.\ X \notin \texttt{FV}\ ([\texttt{rec}\ X\ E'/X]\ E) \qquad \texttt{[NOTIN\_FV\_lemma]}$$

Adding the above lemma (taking $E' = E$) into the assumption list immediately causes a contradiction with assumption 4, and the proof finally completes. □

Without the above result, the proof of Milner's "unique solution of equations" theorem (for $\sim$) cannot complete. Compared with the above proof, the proof of the similar result for bound variables is easier:

Proposition 5.2. *if* $E \xrightarrow{\mu} E'$ *then* $\mathsf{bv}(E') \subseteq \mathsf{bv}(E)$*, or formally:*

$$\vdash \forall E\ u\ E'.\ E\ -u\rightarrow\ E' \Longrightarrow \texttt{BV}\ E' \subseteq \texttt{BV}\ E \qquad \texttt{[TRANS\_BV]}$$

Proof. Following the proof of Prop. 5.1, we do action induction on $E \xrightarrow{\mu} E'$ and simplify all subgoals by set-theoretic facts. The only subgoal left is:

```
      BV E' ⊆ X INSERT BV E
------------------------------------
 0.  BV E' ⊆ BV ([rec X E/X] E)
```

Note the difference with the proof of Prop. 5.1: now we have `INSERT` in place of `DELETE`. Then, we directly use the assumption and the transitivity of $\subseteq$ and try to prove the following goal instead:

```
      BV ([rec X E/X] E) ⊆ X INSERT BV E
------------------------------------
 0.  BV E' ⊆ BV ([rec X E/X] E)
```

This is an easy goal, due to the following lemma which is proved by induction on E:

$$\vdash \forall X\ E\ E'.\ \texttt{BV}\ ([E'/X]\ E) \subseteq \texttt{BV}\ E \cup \texttt{BV}\ E' \qquad \texttt{[BV\_SUBSET]}$$

Taking $E' = \texttt{rec}\, X.\, E$ in the above lemma, we have $\texttt{BV}\ ([\texttt{rec}\ X\ E/X]\ E) \subseteq \texttt{BV}\ E \cup \texttt{BV}\ (\texttt{rec}\ X\ E)$, the right side equals to $\texttt{BV}\ E \cup (X\ \texttt{INSERT}\ \texttt{BV}\ E) = X\ \texttt{INSERT}\ \texttt{BV}\ E$. Thus the proof can be completed. □

5.2. Multivariable substitutions

There are two ways to represent the multivariable substitution of $E\{\widetilde{P}/\widetilde{X}\}$ which substitutes each possibly occurrences of free variables X_i in E with the corresponding P_i: (1) iterately applying the existing univariable `CCS_Subst`; (2) defining a new multivariable substitution function. Note that there is the possibility that $\widetilde{P}$ syntactically again contains variables from $\widetilde{X}$, thus different orders of iterated substitutions may lead to different results. To obtain the maximal flexibility with a reduced effort, we defined a multivariable version of `CCS_Subst` called `CCS_SUBST` based HOL's `finite_map` theory [22].

A finite map of type $\alpha \mapsto \beta$ is like a function of type $\alpha \to \beta$ having only finitely many elements in its domain. In HOL, an empty finite map is denoted as `FEMPTY`. If *fm* is a finite map, its domain (as a set of keys) is denoted as `FDOM` *fm*. Applying *fm* on a certain key, say k, is denoted by *fm* `'` k.

The function `CCS_SUBST` takes a finite map *fm* of type $\alpha \mapsto (\alpha, \beta)$ `CCS` and a CCS expression, and then returns another CCS expression in which all occurrences of variables in the *fm*'s domain are substituted with the corresponding value in *fm*. Such a finite map can be built from the list of variables *Xs* and the corresponding targets *Ps* (of the same length) by a helper function `fromList` (whose details are not interesting and omitted here), while `CCS_SUBST` (`fromList` *Xs Ps*) E is abbreviated as $[Ps/Xs]\ E$. The substitution process is variable-parallel thus order-independent. For most CCS operators it just recursively calls itself on subterms. The only interesting cases are at agent variables and recursion:

$$\texttt{CCS\_SUBST}\ fm\ (\texttt{var}\ X) \stackrel{\mathsf{def}}{=} \textbf{if}\ X \in \texttt{FDOM}\ fm\ \textbf{then}\ fm\ \texttt{'}\ X\ \textbf{else}\ \texttt{var}\ X \qquad \texttt{[CCS\_SUBST\_var]}$$

$$\texttt{CCS\_SUBST}\ fm\ (\texttt{rec}\ X\ E) \stackrel{\mathsf{def}}{=} \textbf{if}\ X \in \texttt{FDOM}\ fm\ \textbf{then}\ \texttt{rec}\ X\ (\texttt{CCS\_SUBST}\ (fm \setminus X)\ E)\ \textbf{else}\ \texttt{rec}\ X\ (\texttt{CCS\_SUBST}\ fm\ E) \qquad \texttt{[CCS\_SUBST\_rec]}$$

Note that the variable substitution only occurs on free variables. In the case of `rec` $X.\,E$: if X (a bound variable) is in the domain of fm, `CCS_SUBST` must continue the substitution on E using a reduced map — without X — so that all occurences of X in E are correctly bypassed. Since `CCS_SUBST` only runs once on each subterm, the possible free variables in Ps are never substituted. Below we present some key lemmas about `CCS_SUBST`, omitting the proofs:[7]

Lemma 5.3 (`CCS_SUBST_elim`). *If the free variables of E is disjoint with Xs, a substitution of Xs in E does not change E:*

$$\vdash \texttt{DISJOINT}\ (\texttt{FV}\ E)\ (\texttt{set}\ Xs) \wedge \texttt{LENGTH}\ Ps = \texttt{LENGTH}\ Xs \Longrightarrow [Ps/Xs]\ E = E$$

Lemma 5.4 (`CCS_SUBST_self`). *Substituting each free variable X in E to* `var` *X (itself) does not change E:*

$$\vdash \texttt{ALL\_DISTINCT}\ Xs \Longrightarrow [\texttt{MAP var}\ Xs/Xs]\ E = E$$

The next lemma plays an important role. It essentially swaps the order of two substitutions:

Lemma 5.5 (`CCS_SUBST_nested`). *Under certain conditions (to get rid of substitution orders), two nested substitutions can be converted into a single substitution where the targets are substituted first:*

$$\begin{array}{l}\vdash \texttt{ALL\_DISTINCT}\ Xs \wedge \texttt{LENGTH}\ Ps = \texttt{LENGTH}\ Xs \wedge \texttt{LENGTH}\ Es = \texttt{LENGTH}\ Xs \wedge \\ \quad \texttt{DISJOINT}\ (\texttt{BV}\ C)\ (\texttt{set}\ Xs) \Longrightarrow \\ \quad [Ps/Xs]\ ([Es/Xs]\ C) = [\texttt{MAP}\ [Ps/Xs]\ Es/Xs]\ C\end{array}$$

The next three lemmas show the relative correctness of `CCS_SUBST` w.r.t. `CCS_Subst`:

Lemma 5.6 (`CCS_SUBST_sing`). *If there is only one single variable X in the map (with the target expression E'), then* `CCS_SUBST` *behaves exactly the same as (the univariable)* `CCS_Subst`*:*

$$\vdash [[E']/[X]]\ E = [E'/X]\ E$$

Lemma 5.7 (`CCS_SUBST_reduce`). *Under certain conditions (to get rid of substitution orders), a multivariable substitution of variables $X::Xs$ can be reduced to a multivariable substitution of variables Xs and an univariable substitution of X:*

$$\begin{array}{l}\vdash \neg\texttt{MEM}\ X\ Xs \wedge \texttt{ALL\_DISTINCT}\ Xs \wedge \texttt{LENGTH}\ Ps = \texttt{LENGTH}\ Xs \wedge \\ \quad \texttt{EVERY}\ (\lambda e.\ X \notin \texttt{FV}\ e)\ Ps \Longrightarrow \\ \quad \forall E.\ [P::Ps/X::Xs]\ E = [P/X]\ ([Ps/Xs]\ E)\end{array}$$

Lemma 5.8 (`CCS_SUBST_FOLDR`). *Under certain conditions (to get rid of substitution orders), a multivariable substitution can be reduced to repeated applications of univariable substitutions of each variable:*

$$\begin{array}{l}\vdash \texttt{ALL\_DISTINCT}\ Xs \wedge \texttt{LENGTH}\ Ps = \texttt{LENGTH}\ Xs \wedge \\ \quad \texttt{EVERY}\ (\lambda(x,p).\ \texttt{FV}\ p \subseteq \{x\})\ (\texttt{ZIP}\ (Xs,Ps)) \Longrightarrow \\ \quad [Ps/Xs]\ E = \texttt{FOLDR}\ (\lambda(x,y)\ e.\ [y/x]\ e)\ E\ (\texttt{ZIP}\ (Xs,Ps))\end{array}$$

Finally, the following two lemmas precisely estimate the free and bound variables of a substituted term:

Lemma 5.9 (`BV_SUBSET_BIGUNION`). *The bound variables of $[Ps/Xs]\ E$ have at most the union of all bound variables from E and Ps.*

[7] Hereafter, some new logical constants from HOL's `pred_set` and `list` theories are used (c.f. [22] for more details.): `DISJOINT` denotes set disjointness; "`set` Xs" is the set converted from the list Xs; `ALL_DISTINCT` says all elements of a list are distinct with each other; `MAP` is the mapping function from one list to another; `EVERY` means each element of a list satisfies a predicate; `ZIP` creates a new list from two lists of the same length, and each element of the new list is a pair of elements, each from one of the old lists.

$\vdash$ ALL_DISTINCT Xs $\wedge$ LENGTH Ps = LENGTH Xs $\wedge$ DISJOINT (BV E) (set Xs) $\Longrightarrow$
BV ($[Ps/Xs]$ E) $\subseteq$ BV E $\cup$ BIGUNION (IMAGE BV (set Ps))

Lemma 5.10 (FV_SUBSET_BIGUNION_PRO)**.** *The free variables of* $[Ps/Xs]$ E *have at most the union of all free variables from* E *and* Ps*, excluding* Xs*.*

$\vdash$ ALL_DISTINCT Xs $\wedge$ LENGTH Ps = LENGTH Xs $\wedge$ DISJOINT (BV E) (set Xs) $\Longrightarrow$
FV ($[Ps/Xs]$ E) $\subseteq$ FV E DIFF set Xs $\cup$ BIGUNION (IMAGE FV (set Ps))

In the above lemmas, the condition DISJOINT (BV E) (set Xs) (the bound variables of E and the free substitution variables are disjoint) is not necessary but makes some proofs (much) easier. As these lemmas are used in the rest work, such disjointness conditions are propagated everywhere.

5.3. Multivariable (weakly guarded) contexts

Recall in Section 4.9, univariable contexts and their guarded companions are defined as predicates over λ-expressions of type $CCS \to CCS$. These predicates, such as CONTEXT (contexts/expressions), WG (weak guarded contexts), SG (guarded contexts) and SEQ (sequential contexts), are all defined inductively, i.e. they are built from some "holes" in a bottom-up way. For instance, WG ($\lambda\, t.\ a.t \mid P$) holds because WG ($\lambda\, t.\ a.t$) holds as a base case of the inductive definition of WG:

$\vdash \forall\, a.$ WG ($\lambda\, t.\ a.t$) [WG1]

For any agent variable X, we have by β-reduction: ($\lambda\, t.\ l.t + P$) (var X) = l.var $X + P$ (or "$l.\, X + P$" in textbook notation) is weakly guarded (c.f. Def. 3.3).

It is possible to inductively assert multivariable contexts and their guarded variants, but care is needed as we need to exclude the cases where free variables occur inside a recusion operator. A more elegant solution is to define multivariable contexts (and variants) upon the existing univariable definitions, which is the approach we have followed. For example, to see the weak guardedness of $a.\, X + b.\, X + c.\, Y$, it is natural to focus on each (free) variable: if we substitute each occurrence of just one variable, say X, with a "hole" and see the resulting term as a λ-function, then all such λ-functions must satisfy WG, i.e. the following results hold:

$\vdash$ WG ($\lambda\, t.\ a.t\ +\ b.t\ +\ c.$var Y)
$\vdash$ WG ($\lambda\, t.\ a.$var $X\ +\ b.$var $X\ +\ c.t$)

Also note that $a.t + b.t + c.$var $Y = [t/X]$ ($a.$var $X + b.$var $X + c.$var Y) can be expressed as a univariable substitution of the original term. This idea leads to the following definitions:

context Xs E $\stackrel{\text{def}}{=}$ EVERY ($\lambda\, X.$ CONTEXT ($\lambda\, t.$ $[t/X]$ E)) Xs [context_def]

weakly_guarded Xs E $\stackrel{\text{def}}{=}$ EVERY ($\lambda\, X.$ WG ($\lambda\, t.$ $[t/X]$ E)) Xs [weakly_guarded_def]

Notice that, the above definitions take an extra list of variables Xs and assert the CCS expression E with respect to this list. This allows us to formalise the concepts of contexts and guardedness independently of FV. Then weakly_guarded (SET_TO_LIST (FV E)) E can be used assert that E is weakly guarded w.r.t. all its free variables.[8] (The guardedness and sequentiality can also be defined similarily when needed, using their univariable companions (SG and SEQ).)

The most important property of contexts are that, strong bisimilarity and other (pre)congruence relations are preserved by contexts, e.g.

[8] Here SET_TO_LIST converts a finite set to a list of the same elements. It turns out that, we never need this, because in all unique-solution theorems a list of variables Xs is fixed and then all equations are required to contain free variables up to Xs. Thus our choices in these definitions are optimal.

Lemma 5.11 (STRONG_EQUIV_subst_context). *If two tuples of processes $\widetilde{P}$ and $\widetilde{Q}$ are strongly bisimilar*[9] *for any context E with the possible occurrences of variables $\widetilde{X}$ (same length as $\widetilde{P}$ and $\widetilde{Q}$), $E\{\widetilde{P}/\widetilde{X}\}$ and $E\{\widetilde{Q}/\widetilde{X}\}$ are also strongly bisimilar:*

$\vdash$ ALL_DISTINCT Xs $\wedge$ LENGTH Ps = LENGTH Xs $\wedge$ $Ps \sim Qs$ $\Longrightarrow$
$\forall E$. context Xs E $\Longrightarrow$ $[Ps/Xs]$ $E \sim [Qs/Xs]$ E

The similar properties also hold if $\sim$ is replaced with rooted bisimilarity ($\approx^{c}$) or rooted contraction ($\succeq^{c}_{bis}$). (It does not hold for weak bisimilarity $\approx$ and the contraction preorder $\succeq_{bis}$.)

Another important property of contexts is their composability: if we substitute some free variables in a context for some other contexts, the resulting term is still a context (w.r.t. the same set of variables):

$\vdash$ ALL_DISTINCT Xs $\wedge$ context Xs C $\wedge$ EVERY (context Xs) Es $\wedge$
LENGTH Es = LENGTH Xs $\Longrightarrow$
context Xs $([Es/Xs]$ $C)$ [context_combin]

Not every expression fit with CCS syntax is a context: any free variable in concern must not occur inside any recusion operator (c.f. the footnote at the beginning of Section 3.1). In the extreme case, if the set of free variables of an expression, say E, is disjoint with a list of variables $\widetilde{X}$, then this expression is for sure a context:

$\vdash$ DISJOINT (FV E) (set Xs) $\Longrightarrow$ context Xs E [disjoint_imp_context]

On the other hand, for any context (w.r.t. $\widetilde{X}$) of the form $\mathtt{rec}\, Y.\, E$, we can conclude that, the set of free variables of E excluding Y, is disjoint with $\widetilde{X}$, i.e.

$\vdash$ context Xs (rec Y E) $\Longrightarrow$ DISJOINT (FV E DELETE Y) (set Xs) [context_rec]

Notice that, the above result is the best possible one. We cannot conclude DISJOINT (FV E) (set Xs), because Y as a bound variable of $\mathtt{rec}\, Y.\, E$ may be also a free variable in $\widetilde{X}$. We also cannot conclude context Xs E, because in $\mathtt{rec}\, Y.\, E$ it is possible that the bound variable Y occurs inside another recursion operator, then in E it now becomes free. (See $\mathtt{rec}\, Y.\, (\mathtt{rec}\, Z.\, (a.\, Y + b.\, Z) + c.\, Y)$ for such an example.)

For weakly-guarded contexts (which are also normal contexts), beside their regular properties (i.e. weak guardedness) as in the univariable case (WG), we also need their composability w.r.t. multivariable contexts: if we substitute some free variables in a context C for some weakly-guarded contexts, the resulting term is weakly-guarded (w.r.t. the same set of variables):

$\vdash$ ALL_DISTINCT Xs $\wedge$ context Xs C $\wedge$ weakly_guarded Xs Es $\wedge$
LENGTH Es = LENGTH Xs $\Longrightarrow$
weakly_guarded Xs $([Es/Xs]$ $C)$ [weakly_guarded_combin]

5.4. Multivariable equations and solutions

With the formal definitions of multivariable substitution and multivariable (weakly-guarded) contexts, now we are ready to formally define multivariable CCS equations/contractions and their (unique) solutions. Consider a system of equation $\{X_i = E_i\}_{i \in I}$ (Def. 3.1), or its expanded form (here we suppose $I = [1, n] \in \mathbb{N}$, i.e. a finite list):

$$\begin{cases} X_1 = E_1[\widetilde{X}] \\ X_2 = E_2[\widetilde{X}] \\ \dots \\ X_n = E_n[\widetilde{X}] \end{cases}$$

Consider its two essential ingredients:

[9] Hereafter, HOL terms like $Ps \sim Qs$ means that, two list of CCS processes are componentwise (strongly) bisimilar. The same notation $\sim$ has been overloaded.

- $\widetilde{X} = (X_1, X_2, \ldots, X_n)$: a list of equation variables;
- $\widetilde{E} = (E_1, E_2, \ldots, E_n)$: a list of CCS contexts with possible occurrences of free variables in $\widetilde{X}$.

Obviously the two lists must have the same length. The all distinctness of $\widetilde{X}$ is not really necessary but is very reasonable. Furthermore, we should assume $\widetilde{E}$ does not contain any other free variables not in $\widetilde{X}$, as Milner's unique-solution theorems also require the same condition.[10]

In this project, the only optional restriction we added here is that, for each E_i the set of its bound variables must be *disjoint* with $\widetilde{X}$. This restriction is actually aligned with literature based on process constants as one should never use the same name for equation variables and constants. However, we believe this restriction can be removed in the future. [11] Putting all together, below is the fomal definition of multivariable CCS equation:

$$\vdash \texttt{CCS\_equation}\ Xs\ Es \iff \texttt{ALL\_DISTINCT}\ Xs \wedge \texttt{LENGTH}\ Es = \texttt{LENGTH}\ Xs \wedge \texttt{EVERY}\ (\lambda e.\ \texttt{FV}\ e \subseteq \texttt{set}\ Xs)\ Es \wedge \texttt{EVERY}\ (\lambda e.\ \texttt{DISJOINT}\ (\texttt{BV}\ e)\ (\texttt{set}\ Xs))\ Es \quad \texttt{[CCS\_equation\_def]}$$

Now consider (formally) what is a solution $\widetilde{P}$ of CCS equation. First of all, the definition should be parametrized on a binary CCS relation $\mathcal{R}$ such as $\sim$ and $\succeq^{\mathrm{c}}_{\mathrm{bis}}$, so that the single definition could be used for the solution of all kind of CCS equations/contractions. Then, of course $\widetilde{P}\ \mathcal{R}\ \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$ must hold, i.e.

$$\begin{cases} P_1\ \mathcal{R}\ E_1\{\widetilde{P}/\widetilde{X}\} \\ P_2\ \mathcal{R}\ E_2\{\widetilde{P}/\widetilde{X}\} \\ \ldots \\ P_n\ \mathcal{R}\ E_n\{\widetilde{P}/\widetilde{X}\} \end{cases}$$

Furthermore, each P_i should be pure process, i.e. having no free variable. This is usually implicitly assumed in the literature but is actually a *must* to complete some unique-solution proofs. At last, like the case of CCS equations, we must assume that, the set of bound variables (i.e. process constants) must be disjoint with $\widetilde{X}$. (This disjointness requirement is optional but makes many proofs much easier, and it can be removed in the future.) Putting all together, below is the formal definition of a solution of multivariable CCS equations:[12]

$$\vdash \texttt{CCS\_solution}\ R\ Xs\ Es\ Ps \iff \texttt{ALL\_PROC}\ Ps \wedge \texttt{EVERY}\ (\lambda e.\ \texttt{DISJOINT}\ (\texttt{BV}\ e)\ (\texttt{set}\ Xs))\ Ps \wedge \texttt{LIST\_REL}\ R\ Ps\ (\texttt{MAP}\ [Ps/Xs]\ Es) \quad \texttt{[CCS\_solution\_def]}$$

Notice that, in HOL, $\texttt{CCS\_solution}\ R\ Xs\ Es\ Ps \iff Ps \in \texttt{CCS\_solution}\ R\ Xs\ Es$. The form in right side suggest that, $\texttt{CCS\_solution}\ R\ Xs\ Es$ is actually a set containing all solutions (of $\texttt{CCS\_equation}\ Xs\ Es$). Then the unique-solution theorems can be understood as: any two (syntactically different) elements in the set are bisimilar.

[10] One may think that, if all other (i.e. not in $\widetilde{X}$) free variables in $\widetilde{E}$ have the same functionalities as $\mathbf{0}$ as they are not to be substituted, then all versions of unique-solution theorems should still hold. However this is not the case for at least Milner's Theorem 3.4 as we shall explain.

[11] Essentially, this restriction is due to the proof difficulties in some early lemmas about multivariable substitutions where $\texttt{DISJOINT}\ (\texttt{BV}\ E)\ (\texttt{set}\ Xs)$ unnecessarily appears in the antecedents. Removing it requires a more complex proof but is indeed possible. (We have succeeded in fixing many such lemmas.) Once all these lemmas are fixed, the final unique-solution theorems also do not need this disjointness restriction.

[12] If R is a binary relation of CCS processes, $\texttt{LIST\_REL}\ R$ is the same binary relation but for lists of CCS processes. Furthermore, $\texttt{LIST\_REL}\ R\ A\ B$ implicitly implies that the two lists A and B have the same length.

5.5. Unique solution of equations/contractions (the multivariable version)

With all above new devices (multivariable contexts and substitutions), the multivariable case of Lemma 3.15 is formalised below:

$$\begin{array}{l}\vdash \texttt{weakly\_guarded}\ Xs\ E \wedge \texttt{FV}\ E \subseteq \texttt{set}\ Xs \wedge \texttt{DISJOINT}\ (\texttt{BV}\ E)\ (\texttt{set}\ Xs) \Longrightarrow \\ \quad \forall\, Ps\,. \\ \qquad \texttt{LENGTH}\ Ps = \texttt{LENGTH}\ Xs \Longrightarrow \\ \qquad \forall\, u\ P'. \\ \qquad\quad [Ps/Xs]\ E\ -u\rightarrow\ P' \Longrightarrow \\ \qquad\quad \exists\, E'. \\ \qquad\qquad \texttt{context}\ Xs\ E' \wedge \texttt{FV}\ E' \subseteq \texttt{set}\ Xs \wedge \texttt{DISJOINT}\ (\texttt{BV}\ E')\ (\texttt{set}\ Xs) \wedge \\ \qquad\qquad P' = [Ps/Xs]\ E' \wedge \\ \qquad\qquad \forall\, Qs.\ \texttt{LENGTH}\ Qs = \texttt{LENGTH}\ Xs \Longrightarrow [Qs/Xs]\ E\ -u\rightarrow\ [Qs/Xs]\ E'\end{array}$$

Notice that, while Milner did not pointed out explicitly, the involved weakly guarded context E must contain free variables up to $\widetilde{X}$ — the same requirement as in Theorem 3.4, which otherwise does not complete. Besides, we further required that, the bound variables of E must be disjoint with $\widetilde{X}$.

The multivariable version of Theorem 3.4 is formalised below:

$$\begin{array}{l}\vdash \texttt{CCS\_equation}\ Xs\ Es \wedge \texttt{weakly\_guarded}\ Xs\ Es \wedge \\ \quad Ps \in \texttt{CCS\_solution STRONG\_EQUIV}\ Xs\ Es \wedge \\ \quad Qs \in \texttt{CCS\_solution STRONG\_EQUIV}\ Xs\ Es \Longrightarrow \\ \quad Ps \sim Qs \qquad\qquad \texttt{[strong\_unique\_solution\_thm]}\end{array}$$

And the multivariable version of Theorem 3.16:

$$\begin{array}{l}\vdash \texttt{CCS\_equation}\ Xs\ Es \wedge \texttt{weakly\_guarded}\ Xs\ Es \wedge \\ \quad Ps \in \texttt{CCS\_solution OBS\_contracts}\ Xs\ Es \wedge \\ \quad Qs \in \texttt{CCS\_solution OBS\_contracts}\ Xs\ Es \Longrightarrow \\ \quad Ps \approx^{\mathrm{c}} Qs \qquad\qquad \texttt{[unique\_solution\_of\_rooted\_contractions]}\end{array}$$

Compared with their formal proofs in the univariable case, although each step related to multivariable substitutions is more difficult than the univariable case, the entire proofs still follow the same outlines, and many proof scripts can be directly copied from their univariable version.

6. Related work on formalisation

Monica Nesi did the first CCS formalisations for both pure and value-passing CCS [15, 27] using early versions of the HOL theorem prover.[13] Her main focus was on implementing decision procedures (as a ML program, e.g. [28]) for automatically proving bisimilarities of CCS processes. Her work has been a basis for ours [25]. However, the differences are substantial, especially in the way of defining bisimilarities. We greatly benefited from new 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 [29, 30]. More details can be found in Bengtson's PhD thesis [31]. For CCS, he has formalized basic properties for strong/weak equivalence (congruence, basic algebraic laws); however the CCS syntax doesn't have constants (or recursion operator), using replication in place of recursion. Other formalisations in this area include the early work of T.F. Melham [32] and O.A. Mohamed [33] in HOL, Compton [34] in Isabelle/HOL, Solange[14] in Coq and Chaudhuri et al. [35] in Abella, the latter focuses on 'bisimulation up-to' techniques (for strong bisimilarity) for CCS and π-calculus. Damien Pous [36] also formalised up-to techniques and some CCS examples in Coq. Formalisations less related to ours include Kahsai and Miculan [37] for the spi calculus in Isabelle/HOL, and Hirschkoff [38] for the π-calculus in Coq.

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

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

7. Conclusions and future work

In this paper, beside the introduction of rooted contraction and its unique solution theorems as a theoretical extension of [8], we highlighted a comprehensive formalisation of the theory of CCS in the HOL4 theorem prover. In particular, the formalisation supports four methods for establishing (strong and weak) bisimilarities:

1. constructing a bisimulation (the standard bisimulation proof method);
2. constructing a 'bisimulation up-to';
3. employing algebraic laws;
4. constructing 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, both univariable and multivariable cases. It has 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-prone, especially in cases where the proofs are similar to each other or when there are long case analyses to be carried out. Some of the textbook proofs were even wrong and the correct version is known but not available in the literature. This is the case of Milner's proof of the unique-solution theorem for wb (Theorem 3.5). Now we have fully verified formal proofs for people who can read them. For some other textbook proofs, even they are correct, sometimes they actually do not need all information from the antecedents, which can be further weakened. This kind of improvements can be easily found during the formalisation work. For our CCS formalisation, we believe that all the definitions and theorem statements are easy to read and understand, as they are very close to their original statements in textbooks [2, 1].

For the future work, formalising other equivalences and preorders could also be considered, notably the trace-based 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 [39] to unique solutions, in which the use of contraction is replaced by semantic conditions on process transitions such as divergence. Finally, the disjointness between free and bound variables in the current multivariable formalisation is certainly a limitation that can be removed in the future. If needed, extending the current work to infinite number of equations and variables is also feasible.

Acknowledgements. We have benefitted from suggestions and comments from the anonymous reviewers and several people from the HOL community, including (in alphabetic 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.

[1] R. Milner, Communication and concurrency, PHI Series in computer science, Prentice-Hall, 1989.
[2] R. Gorrieri, C. Versari, Introduction to Concurrency Theory, Transition Systems and CCS, Springer, Cham, 2015. doi:10.1007/978-3-319-21491-7.
[3] 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. doi:10.4204/EPTCS.276.10.
[4] J. C. M. Baeten, T. Basten, M. A. Reniers, Process Algebra: Equational Theories of Communicating Processes, Cambridge University Press, 2010. doi:10.1017/CBO9781139195003.
[5] A. W. Roscoe, The theory and practice of concurrency, Prentice Hall, 1998.
URL http://www.cs.ox.ac.uk/people/bill.roscoe/publications/68b.pdf
[6] A. W. Roscoe, Understanding Concurrent Systems, Springer, 2010. doi:10.1007/978-1-84882-258-0.
[7] D. Sangiorgi, Equations, contractions, and unique solutions, in: ACM SIGPLAN Notices, Vol. 50, ACM, 2015, pp. 421–432. doi:10.1145/2676726.2676965.
URL https://hal.inria.fr/hal-01089205
[8] D. Sangiorgi, Equations, contractions, and unique solutions, ACM Transactions on Computational Logic (TOCL) 18 (2017) 4. doi:10.1145/2971339.
URL https://hal.inria.fr/hal-01647063

[9] 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. doi:10.1007/11601548_4.
[10] R. Milner, Operational and algebraic semantics of concurrent processes, Handbook of Theoretical Comput. Sci.
[11] D. Sangiorgi, Introduction to Bisimulation and Coinduction, Cambridge University Press, 2011. doi:10.1017/CBO9780511777110.
[12] S. Arun-Kumar, M. Hennessy, An efficiency preorder for processes, Acta Informatica 29 (8) (1992) 737–760. doi:10.1007/BF01191894.
[13] M. J. C. Gordon, T. F. Melham, Introduction to HOL: A Theorem Proving Environment for Higher Order Logic, Cambrige University Press, New York, NY, USA, 1993.
[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. doi:10.1007/978-3-540-71067-7_6.
URL http://ts.data61.csiro.au/publications/nicta_full_text/1482.pdf
[15] 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
[16] HOL4 contributors, The HOL System LOGIC (Kananaskis-13 release) (Aug. 2019).
URL http://sourceforge.net/projects/hol/files/hol/kananaskis-13/kananaskis-13-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. 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. doi:10.2307/2266170.
[20] T. F. Melham, Automating Recursive Type Definitions in Higher Order Logic, in: Current Trends in Hardware Verification and Automated Theorem Proving, Springer New York, New York, NY, 1989, pp. 341–386.
[21] T. F. Melham, A Package For Inductive Relation Definitions In HOL, in: 1991., International Workshop on the HOL Theorem Proving System and Its Applications, IEEE, 1991, pp. 350–357.
[22] HOL4 contributors, The HOL System DESCRIPTION (Kananaskis-13 release) (Aug. 2019).
URL http://sourceforge.net/projects/hol/files/hol/kananaskis-13/kananaskis-13-description.pdf
[23] R. Gorrieri, CCS (25, 12) is Turing-complete, Fundamenta Informaticae 154 (1-4) (2017) 145–166. doi:10.3233/FI-2017-1557.
[24] D. Sangiorgi, R. Milner, The problem of "weak bisimulation up to", in: International Conference on Concurrency Theory, Springer, 1992, pp. 32–46. doi:10.1007/BFb0084781.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.5964
[25] 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/
[26] 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. doi:10.1007/978-3-642-39634-2_12.
[27] M. Nesi, Formalising a Value-Passing Calculus in HOL, Formal Aspects of Computing 11 (2) (1999) 160–199. doi:10.1007/s001650050046.
[28] 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. doi:10.1145/151646.151648.
[29] 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. doi:10.1016/j.entcs.2007.08.017.
[30] J. Parrow, J. Bengtson, Formalising the pi-calculus using nominal logic, Logical Methods in Computer Science 5. doi:10.2168/LMCS-5(2:16)2009.
[31] J. Bengtson, Formalising process calculi, Ph.D. thesis, Acta Universitatis Upsaliensis (2010).
[32] T. F. Melham, A Mechanized Theory of the Pi-Calculus in HOL, Nord. J. Comput. 1 (1) (1994) 50–76.
URL http://core.ac.uk/download/pdf/22878407.pdf
[33] 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. doi:10.1007/3-540-60275-5_53.
[34] M. Compton, Embedding a fair CCS in Isabelle/HOL, in: Theorem Proving in Higher Order Logics: Emerging Trends Proceedings, 2005, p. 30. doi:10.1.1.105.834.
URL https://web.comlab.ox.ac.uk/techreports/oucl/RR-05-02.pdf#page=36
[35] 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. doi:10.1145/2676724.2693170.
[36] D. Pous, New up-to techniques for weak bisimulation, Theoretical Computer Science 380 (2007) 164–180. doi:10.1016/j.tcs.2007.02.060.
[37] T. Kahsai, M. Miculan, Implementing spi calculus using nominal techniques, in: Conference on Computability in Europe, Springer, 2008, pp. 294–305. doi:10.1007/978-3-540-69407-6_33.
[38] 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. doi:10.1007/BFb0028392.

[39] 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. doi:10.4230/LIPIcs.CONCUR.2017.11.
URL http://drops.dagstuhl.de/opus/volltexte/2017/7784