Weak CCP bisimilarity with strong procedures

Luis Pino, Andres Aristizabal, Filippo Bonchi, Frank Valencia

Weak CCP Bisimilarity with Strong Procedures[☆]

Luis F. Pino

INRIA/DGA and LIX (UMR 7161 X-CNRS), École Polytechnique, 91128 Palaiseau Cedex, France

Andrés Aristizábal

Pontificia Universidad Javeriana Cali

Filippo Bonchi

ENS Lyon, Université de Lyon, LIP (UMR 5668 CNRS ENS Lyon UCBL INRIA), 46 Allée d'Italie, 69364 Lyon, France

Frank Valencia

CNRS and LIX (UMR 7161 X-CNRS), École Polytechnique, 91128 Palaiseau Cedex, France

Abstract

Concurrent constraint programming (CCP) is a well-established model for concurrency that singles out the fundamental aspects of asynchronous systems whose agents (or processes) evolve by posting and querying (partial) information in a global medium. Bisimilarity is a standard behavioral equivalence in concurrency theory. However, only recently a well-behaved notion of bisimilarity for CCP, and a CCP partition refinement algorithm for deciding the strong version of this equivalence have been proposed. Weak bisimilarity is a central behavioral equivalence in process calculi and it is obtained from the strong case by taking into account only the actions that are observable in the system. Typically, the standard partition refinement can also be used for deciding weak bisimilarity simply by us-

[☆]This work has been partially supported by the projects ANR-09-BLAN-0345 CPP, ANR 12IS02001 PACE and ANR-09-BLAN-0169-01 PANDA; and by the French Defense procurement agency (DGA) with a PhD grant.

Email addresses: luis.pino@lix.polytechnique.fr (Luis F. Pino), aaaristizabal@javerianacali.edu.co (Andrés Aristizábal), filippo.bonchi@ens-lyon.fr (Filippo Bonchi), frank.valencia@lix.polytechnique.fr (Frank Valencia)

ing Milner's reduction from weak to strong bisimilarity; a technique referred to as *saturation*. In this paper we demonstrate that, because of its involved labeled transitions, the above-mentioned saturation technique does not work for CCP. We give an alternative reduction from weak CCP bisimilarity to the strong one that allows us to use the CCP partition refinement algorithm for deciding this equivalence.

Keywords: Concurrent Constraint Programming, Weak bisimilarity, Partition refinement

1. Introduction

Since the introduction of process calculi, one of the richest sources of foundational investigations stemmed from the analysis of *behavioral equivalences*: in any formal process language, systems which are syntactically different may denote the same process, i.e., they have the same *observable behavior*.

A major dichotomy among behavioral equivalences concerns *strong* and *weak* equivalences. In strong equivalences, all the transitions performed by a system are deemed observable. In weak equivalences, instead, internal transitions (usually denoted by τ) are unobservable. On the one hand, weak equivalences are more abstract (and thus closer to the intuitive notion of behavior); on the other hand, strong equivalences are usually much easier to be checked (for instance, in [1] a strong equivalence is introduced which is computable for a Turing complete formalism).

Strong bisimilarity is one of the most studied behavioral equivalence and many algorithms (e.g., [2, 3, 4]) have been developed to check whether two systems are equivalent up to strong bisimilarity. Among these, the *partition refinement algorithm* [5, 6] is one of the best known: first it generates the state space of a labeled transition system (LTS), i.e., the set of states reachable through the transitions; then, it creates a partition equating all states and afterwards, iteratively, refines these partitions by splitting non equivalent states. At the end, the resulting partition equates all and only bisimilar states.

Weak bisimilarity can be computed by reducing it to strong bisimilarity. Given a (strong) LTS $\longrightarrow$ labeled with actions $a, b, \tau, \ldots$ one can build a (weak) LTS $\Longrightarrow$ using the following inference rules:

$$\frac{P \xrightarrow{a} Q}{P \overset{a}{\Longrightarrow} Q} \qquad \frac{}{P \overset{\tau}{\Longrightarrow} P} \qquad \frac{P \overset{\tau}{\Longrightarrow} P_1 \overset{a}{\Longrightarrow} Q_1 \overset{\tau}{\Longrightarrow} Q}{P \overset{a}{\Longrightarrow} Q}$$

Since weak bisimilarity on $\stackrel{a}{\longrightarrow}$ coincides with strong bisimilarity on $\stackrel{a}{\Longrightarrow}$, then one can check weak bisimilarity with the algorithms for strong bisimilarity on the new LTS $\stackrel{a}{\Longrightarrow}$.

Concurrent Constraint Programming (CCP) [7] is a formalism that combines the traditional algebraic and operational view of process calculi with a declarative one based on first-order logic. In CCP, processes (agents or programs) interact by *adding* (or *telling*) and *asking* information (namely, constraints) in a medium (the *store*). Inspired by [8, 9], the authors introduced in [10] both strong and weak bisimilarity for CCP and showed that the weak equivalence is *fully abstract* with respect to the standard observational equivalence of [11]. Moreover, a variant of the partition refinement algorithm is given in [12] for checking strong bisimilarity on (the finite fragment of) concurrent constraint programming.

In this paper, first we show that the standard method for reducing weak to strong bisimilarity does not work for CCP and then we provide a way out of the impasse. Our solution can be readily explained by observing that the labels in the LTS of a CCP agent are constraints (actually, they are "the minimal constraints" that the store should satisfy in order to make the agent progress). These constraints form a lattice where the least upper bound (denoted by $\sqcup$) intuitively corresponds to conjunction and the bottom element is the constraint $true$. (As expected, transitions labeled by $true$ are internal transitions, corresponding to the τ moves in standard process calculi). Now, rather than closing the transitions just with respect to $true$, we need to close them w.r.t. all the constraints. Formally we build the new LTS with the following rules.

$$\frac{P \stackrel{a}{\longrightarrow} Q}{P \stackrel{a}{\Longrightarrow} Q} \qquad \frac{}{P \stackrel{true}{\Longrightarrow} P} \qquad \frac{P \stackrel{a}{\Longrightarrow} Q \stackrel{b}{\Longrightarrow} R}{P \stackrel{a \sqcup b}{\Longrightarrow} R}$$

The above construction can also be done for CCS [13] by taking sequences of actions $a; b$ rather than $a \sqcup b$. Nevertheless, the resulting transition system may be infinite-branching and hence not amenable to automatic verification using standard algorithms such as partition refinement. In the case of CCP instead, since $\sqcup$ is idempotent, if the original LTS $\stackrel{a}{\longrightarrow}$ has finitely many transitions, then also $\stackrel{a}{\Longrightarrow}$ is finite. This allows us to use the algorithm in [12] to check weak bisimilarity on (the finite fragment) of concurrent constraint programming.

We have developed a tool in which this procedure is implemented. To the best of our knowledge, it is the first tool for checking weak bisimilarity for CCP. It can be found at: `http://www.lix.polytechnique.fr/~andresaristi/checkers/`

Contributions. This paper is an extended version of [14]. In this version we give all the details and proofs omitted in [14]. We generalize some notions, add new examples and extend the intuitions from [14] to improve the clarity of the paper. Furthermore, in Section 3.1.3 we illustrate why our technique would not work for CCS-like process calculi in which two labels cannot be joined into a single one like in CCP. We also add a new section, Section 4, that includes the specification of the algorithm for observational equivalence in CCP, as well as its proof of correctness and complexity. This new section also includes some experiments comparing the strong and weak bisimilarity procedures.

Related Work. CCP is not the only formalism where weak bisimilarity cannot be naively reduced to the strong one. Probably the first case in literature can be found in [2] that introduces an algorithm for checking weak open bisimilarity of π-calculus. This algorithm is rather different from ours, since it is on-the-fly [3] and thus it checks the equivalence of only two given states (while our algorithm, and more generally all algorithms based on partition refinement, check the equivalence of all the states of a given LTS). These algorithms have a polynomial upper bound time complexity. The present algorithm has an exponential time complexity. However, they deal with very different formalisms. Also [15] defines weak labeled transitions following the above-mentioned standard method which does not work in the CCP case.

Analogous problems to the one discussed in this paper arise in Petri nets [16, 17], in tile transition systems [18, 19] and, more generally, in the theory of reactive systems [20, 21, 22] (the interested reader is referred to [23] for an overview). In all these cases, labels form a monoid where the neutral element is the label of internal transitions. In the case of CCS, the fact that a system may perform a transition with a composed label $a; b$ means that it may perform first a transition with a and then a transition with b. This property, which in tile systems [18, 19] is known as vertical decomposition, does not hold for CCP and for the other formalisms mentioned above. As a consequence of this fact, when reducing from weak to strong bisimilarity, one needs to close the transitions with respect to the composition of the monoid (and not only with respect to the neutral element). However, usually, labels composition is not idempotent (as it is for CCP) and thus a finite LTS might be transformed into an infinite one. For this reason, this procedure applied to the afore mentioned cases is not effective for automatic verification.

The result presented in this paper, and more generally our research on CCP, has been largely inspired by the theory of reactive systems by Leifer and Milner [24]: in our operational semantics of CCP, labels represent the "minimal" constraints

that the store should satisfy for executing some transitions, while in [24], labels are the "minimal" contexts.

Structure of the paper. In Section 2 we recall the partition refinement method and the standard reduction from weak to strong bisimilarity. We also recall the CCP formalism, its equivalences, and the CCP partition refinement algorithm. We then show why the standard reduction does not work for CCP. In Section 3 we present our reduction and show its correctness. Finally in Section 4 discuss the decision procedure for observational equivalence in CCP.

2. From Weak to Strong Bisimilarity: Saturation Approach

We start this section by recalling the notion of labeled transition system (LTS), partition and the graph induced by an LTS. Then we present the standard partition refinement algorithm for checking strong bisimilarity, and also how to use it to verify weak bisimilarity. Next we introduce concurrent constraint programming (CCP) including the partition refinement algorithm for CCP. Finally, we shall show that the standard reduction from weak to strong bisimilarity does not work for CCP.

Labeled Transition System. A labeled transition system (LTS) is a triple $(S, L, \rightsquigarrow)$ where S is a set of states, L a set of labels and $\rightsquigarrow \subseteq S \times L \times S$ a transition relation. We shall use $s \overset{a}{\rightsquigarrow} r$ to denote the transition $(s, a, r) \in \rightsquigarrow$. Given a transition $t = (s, a, r)$ we define the source, the target and the label as follows $src(t) = s$, $tar(t) = r$ and $lab(t) = a$. We assume the reader to be familiar with the standard notion of bisimilarity [13].

Partition. Given a set S, a *partition* $\mathcal{P}$ of S is a set of non-empty *blocks*, i.e., subsets of S, that are all disjoint and whose union is S. We write $\{B_1\} \dots \{B_n\}$ to denote a partition consisting of (non-empty) blocks $B_1, \dots, B_n$. A partition represents an equivalence relation where equivalent elements belong to the same block. We write $s\mathcal{P}r$ to mean that s and r are equivalent in the partition $\mathcal{P}$.

LTSs and Graphs. Given a LTS $(S, L, \rightsquigarrow)$, we write $LTS_{\rightsquigarrow}$ for the directed graph whose vertices are the states in S and edges are the transitions in $\rightsquigarrow$. Given a set of initial states $IS \subseteq S$, we write $LTS_{\rightsquigarrow}(IS)$ for the subgraph of $LTS_{\rightsquigarrow}$ rechable from IS. Given a graph G we write $\mathtt{V}(G)$ and $\mathtt{E}(G)$ for the set of vertices and edges of G, respectively.

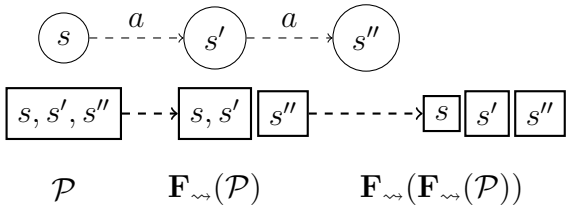

Figure 1: An example of the use of $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$ from Equation 1

2.1. Partition Refinement

We recall the partition refinement algorithm [5] for checking bisimilarity over the states of an LTS $(S, L, \rightsquigarrow)$.

Given a set of initial states $IS \subseteq S$, the partition refinement algorithm (see Algorithm 1) checks bisimilarity on IS as follows. First, it computes $IS^{\star}_{\rightsquigarrow}$, that is the set of all states that are reachable from IS using $\rightsquigarrow$. Then it creates the partition $\mathcal{P}^0$ where all the elements of $IS^{\star}_{\rightsquigarrow}$ belong to the same block (i.e., they are all equivalent). After the initialization, it iteratively refines the partitions by employing the function on partitions $\mathbf{F}_{\rightsquigarrow}(-)$, defined as follows: for a partition $\mathcal{P}$, $s\mathbf{F}_{\rightsquigarrow}(\mathcal{P})r$ iff

$$\text{if } s \overset{a}{\rightsquigarrow} s' \text{ then exists } r' \text{ s.t. } r \overset{a}{\rightsquigarrow} r' \text{ and } s'\mathcal{P}r'. \tag{1}$$

See Figure 1 for an example of $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$. Algorithm 1 terminates whenever two consecutive partitions are equivalent. In such a partition two states (reachable from IS) belong to the same block iff they are bisimilar (using the standard notion of bisimilarity [13]).

Algorithm 1 $\mathtt{pr}(IS, \rightsquigarrow)$

Initialization

1. $IS^{\star}_{\rightsquigarrow}$ is the set of all states reachable from IS using $\rightsquigarrow$,
2. $\mathcal{P}^0 := IS^{\star}_{\rightsquigarrow}$,

Iteration $\mathcal{P}^{n+1} := \mathbf{F}_{\rightsquigarrow}(\mathcal{P}^n)$ as in Equation 1
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

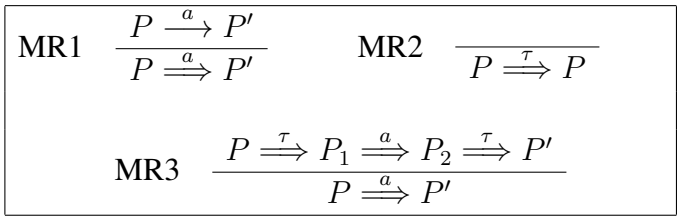

Table 1: Milner's Saturation Method

Standard reduction from weak to strong bisimilarity. As pointed out in the literature (Chapter 3 in [25]), in order to compute weak bisimilarity, we can use the above-mentioned partition refinement. The idea is to start from the LTS generated using the operational semantics ($\longrightarrow$) and then *saturate* it using the rules described in Table 1. Now the problem whether two states s and s' are weakly bisimilar can be reduced to checking whether they are strongly bisimilar w.r.t. $\Longrightarrow$. Formally, we can call $\mathtt{pr}(\{s, s'\}, \Longrightarrow)$ to check whether s and s' are weakly bisimilar. Henceforth, we shall refer to this as *Milner's saturation method*.

As we will show later on, this approach does not work in a formalism like CCP. We shall see that the problem is related to the fact that the transitions in CCP are labeled with constraints, thus they can be arbitrary combined (using the least upper bound operator) to form a new label. Notice that this is not the case for the transitions in CCS-like process calculi. Therefore we will need to define a different saturation method to be able to use the saturation approach for verifying weak bisimilarity in CCP. Namely, we will modify the rules in Table 1 so we can still use the standard strategy for deciding observational equivalence in CCP.

2.2. Concurrent Constraint Programming (CCP)

We shall now recall CCP and the adaptation of the partition refinement algorithm to compute bisimilarity in CCP [12].

2.2.1. Constraint Systems

The CCP model is parametric in a *constraint system (cs)* specifying the structure and interdependencies of the information that processes can ask or and add to a *central shared store*. This information is represented as assertions traditionally referred to as *constraints*. Following [26, 27] we regard a cs as a complete algebraic lattice in which the ordering $\sqsubseteq$ is the reverse of an entailment relation: $c \sqsubseteq d$ means d *entails* c, i.e., d contains "more information" than c. The top

element $false$ represents inconsistency, the bottom element $true$ is the empty constraint, and the *least upper bound* (lub) $\sqcup$ is the join of information.

Definition 1 (cs)**.** *A* constraint system (cs) $\mathbf{C} = (Con, Con_0, \sqsubseteq, \sqcup, true, false)$ *is a complete algebraic lattice where* Con*, the set of constraints, is a partially ordered set wrt* $\sqsubseteq$*,* Con_0 *is the subset of* compact *elements of* Con*,* $\sqcup$ *is the lub operation defined on all subsets, and* $true$*,* $false$ *are the least and greatest elements of* Con*, respectively.*

Recall that $\mathbf{C}$ is a *complete lattice* if every subset of Con has a least upper bound in Con. An element $c \in Con$ is *finite* if for any directed subset D of Con, $c \sqsubseteq \bigsqcup D$ implies $c \sqsubseteq d$ for some $d \in D$. $\mathbf{C}$ is *algebraic* if each element $c \in Con$ is the least upper bound of the finite elements below c.

In order to model *hiding* of local variables and *parameter passing*, in [11] the notion of constraint system is enriched with *cylindrification operators* and *diagonal elements*, concepts borrowed from the theory of cylindric algebras (see [28]).

Let us consider a (denumerable) set of variables Var with typical elements $x, y, z, \ldots$ Define $\exists_{Var}$ as the family of operators $\exists_{Var} = \{\exists_x \mid x \in Var\}$ (*cylindric operators*) and D_{Var} as the set $D_{Var} = \{d_{xy} \mid x, y \in Var\}$ (*diagonal elements*).

A *cylindric constraint system* over a set of variables Var is a constraint system whose underlying support set $Con \supseteq D_{Var}$ is closed under the cylindric operators $\exists_{Var}$ and quotiented by Axioms $\mathrm{C1} - \mathrm{C4}$, and whose ordering $\sqsubseteq$ satisfies Axioms $\mathrm{C5} - \mathrm{C7}$:

C1. $\exists_x \exists_y c = \exists_y \exists_x c$	**C2**. $d_{xx} = true$
C3. *if* $z \neq x, y$ *then* $d_{xy} = \exists_z (d_{xz} \sqcup d_{zy})$	**C4**. $\exists_x (c \sqcup \exists_x d) = \exists_x c \sqcup \exists_x d$
C5. $\exists_x c \sqsubseteq c$	**C6**. *if* $c \sqsubseteq d$ *then* $\exists_x c \sqsubseteq \exists_x d$
C7. *if* $x \neq y$ *then* $c \sqsubseteq d_{xy} \sqcup \exists_x (c \sqcup d_{xy})$	

where c, c_i, d indicate finite constraints, and $\exists_x c \sqcup d$ stands for $(\exists_x c) \sqcup d$. For our purposes, it is enough to think the operator $\exists_x$ as *existential quantifier* and the constraint d_{xy} as the equality $x = y$.

Cylindrification and diagonal elements allow us to model the variable renaming of a formula ϕ; in fact, by the aforementioned axioms, we have that the formula $\exists_x (d_{xy} \sqcup \phi) = true$ can be depicted as the formula $\phi[y/x]$, i.e., the formula obtained from ϕ by replacing all free occurrences of x by y.

We assume notions of *free variable* and of *substitution* that satisfy the following conditions, where $c[y/x]$ is the constraint obtained by substituting x by y in c

and $fv(c)$ is the set of free variables of c: (1) if $y \notin fv(c)$ then $(c[y/x])[x/y] = c$; (2) $(c \sqcup d)[y/x] = c[y/x] \sqcup d[y/x]$; (3) $x \notin fv(c[y/x])$; (4) $fv(c \sqcup d) = fv(c) \cup fv(d)$.

Example 1 (A Constraint System of Linear Order Arithmetic)**.** *Consider the following syntax:*

$$\phi, \psi \ldots := \ t = t' \mid t > t' \mid \phi \ \vee \ \psi \mid \neg\phi$$

where the terms t, t' can be elements of a set of variables Var, or constant symbols $0, 1, \ldots$. Assume an underlying first-order structure of linear-order arithmetic with the obvious interpretation in the natural numbers ω of $=, >$ and the constant symbols.

A variable assignment is a function $\mu : Var \longrightarrow \omega$. We use $\mathcal{A}$ to denote the set of all assignments; $\mathcal{P}(X)$ to denote the powerset of a set X, $\emptyset$ the empty set and $\cap$ the intersection of sets. We use $\mathcal{M}(\phi)$ to denote the set of all assignments that satisfy *the formula ϕ, where the definition of* satisfaction *is as expected.*

We can now introduce a constraint system *as follows: the set of constraints is $\mathcal{P}(\mathcal{A})$, and define $c \sqsubseteq d$ iff $c \supseteq d$. The constraint $false$ is $\emptyset$, while $true$ is $\mathcal{A}$. Given two constraints c and d, $c \sqcup d$ is the intersection $c \cap d$. By abusing the notation, we will often use a formula ϕ to denote the corresponding constraint, i.e., the set of all assignments satisfying ϕ. E.g. we use $x > 1 \sqsubseteq x > 5$ to mean $\mathcal{M}(x > 1) \sqsubseteq \mathcal{M}(x > 5)$. For this constraint system one can show that e is a compact constraint (i.e., e is in Con_0) iff e is a co-finite set in $\mathcal{A}$ (i.e., iff the complement of e in $\mathcal{A}$ is a finite set). For example, $x > 10 \wedge y > 42$ is a compact constraint for $Var = \{x, y\}$.*

From this structure, let us now define the cylindric constraint system $\mathcal{S}$ *as follows. We say that an assignment μ' is* an x-variant *of μ if $\forall y \neq x$, $\mu(y) = \mu'(y)$. Given $x \in Var$ and $c \in \mathcal{P}(\mathcal{A})$, the constraint $\exists_x c$ is the set of assignments μ such that exists $\mu' \in c$ that is an x-variant of μ. The diagonal element d_{xy} is $x = y$.*

Remark 1. *We shall assume that the constraint system is well-founded and, for practical reasons, that its ordering $\sqsubseteq$ is decidable.*

2.2.2. Processes

We now recall the basic CCP process constructions.

Syntax. Let us presuppose a constraint system $\mathbf{C} = (Con, Con_0, \sqsubseteq, \sqcup, true, false)$. The CCP processes are given by the following syntax:

$$P, Q ::= \mathbf{stop} \mid \mathbf{tell}(c) \mid \mathbf{ask}(c) \rightarrow P \mid P \parallel Q \mid P + Q \mid \exists_x P$$

where $c, e \in Con_0$ and $x \in Var$. We use *Proc* to denote the set of all processes.

Finite processes. Intuitively, $\mathbf{stop}$ represents termination, the tell process $\mathbf{tell}(c)$ adds c to the global store. This addition is performed regardless of the generation of inconsistent information. The ask process $\mathbf{ask}(c) \rightarrow P$ may execute P if c is entailed by the information in the store. The processes $P \parallel Q$ and $P + Q$ stand, respectively, for the *parallel execution* and *non-deterministic choice* of P and Q; $\exists_x$ is a *hiding operator*, namely it indicates that in $\exists_x P$ the variable x is *local* to P. The occurrences of x in $\exists_x P$ are said to be bound. The bound variables of P, $bv(P)$, are those with a bound occurrence in P, and its free variables, $fv(P)$, are those with an unbound occurrence[1].

Remark 2. *In this paper we restrict the CCP syntax to finite processes, since we shall deal with infinite-state transition system. Typically, to specify infinite behavior, CCP provides parametric process definitions. A process $p(\vec{z})$ is said to be a procedure call with identifier p and actual parameters $\vec{z}$. For each procedure call $p(z_1 \ldots z_m)$ there exists a unique (possibly recursive) procedure definition of the form $p(x_1 \ldots x_m) \stackrel{\mathtt{def}}{=} P$ where $fv(P) \subseteq \{x_1, \ldots, x_m\}$. Furthermore recursion is required to be* guarded*, i.e., each procedure call within P must occur within an ask process. The behavior of $p(z_1 \ldots z_m)$ is that of $P[z_1 \ldots z_m / x_1 \ldots x_m]$, i.e., P with each x_i replaced with z_i (applying α-conversion to avoid clashes).*

2.2.3. Reduction Semantics

The operational semantics of CCP is given by an *unlabeled* transition relation between configurations. A configuration is a pair $\langle P, d \rangle$ representing a *state* of a system; d is a constraint representing the global store, and P is a process, i.e., a term of the syntax. We use $Conf$ with typical elements $\gamma, \gamma', \ldots$ to denote the set of configurations. A transition $\gamma \longrightarrow \gamma'$ intuitively means that the configuration γ can reduce to γ'. We call these kind of unlabeled transitions *reductions* and we use $\longrightarrow^*$ to denote the reflexive and transitive closure of $\longrightarrow$. Formally, the reduction semantics of CCP is given by the relation $\longrightarrow$ defined in Table 2.

Rules R1, R3 and R4 are easily seen to realize the intuitions described in Section 2.2.2. Rule R2 states that $\mathbf{ask}(c) \rightarrow P$ can evolve to P whenever the global store d entails c. Rule R5 is somewhat more involved. Intuitively, $\exists_x^e P$ behaves like P, except that the variable x possibly present in P must be considered local, and that the information present in e has to be taken into account. [2] It is

[1] Notice that we also defined $fv(.)$ on constraints in the previous section.

[2] Notice that in the syntax we have $\exists_x P$ which refers to $\exists_x^e P$ when $e = true$, namely a local process always starts with an empty local store.

$$\text{R1}\ \langle \mathbf{tell}(c), d\rangle \longrightarrow \langle \mathbf{stop}, d \sqcup c\rangle \qquad \text{R2}\ \frac{c \sqsubseteq d}{\langle \mathbf{ask}\ (c)\ \rightarrow\ P, d\rangle \longrightarrow \langle P, d\rangle}$$

$$\text{R3}\ \frac{\langle P, d\rangle \longrightarrow \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \longrightarrow \langle P' \parallel Q, d'\rangle} \qquad \text{R4}\ \frac{\langle P, d\rangle \longrightarrow \langle P', d'\rangle}{\langle P\ +\ Q, d\rangle \longrightarrow \langle P', d'\rangle}$$

$$\text{R5}\ \frac{\langle P, e \sqcup \exists_x d\rangle \longrightarrow \langle P', e' \sqcup \exists_x d\rangle}{\langle \exists_x^e P, d\rangle \longrightarrow \langle \exists_x^{e'} P', d \sqcup \exists_x e'\rangle}$$

Table 2: Reduction semantics for CCP (symmetric rules for R3 and R4 are omitted).

convenient to distinguish between the *external* and the *internal* points of view. From the internal point of view, the variable x, possibly occurring in the global store d, is hidden. This corresponds to the usual scoping rules: the x in d is *global*, hence "covered" by the local x. Therefore, P has no access to the information on x in d, and this is achieved by filtering d with $\exists_x$. Furthermore, P can use the information (which may also concern the local x) that has been produced locally and accumulated in e. In conclusion, if the visible store at the external level is d, then the store that is visible internally by P is $e \sqcup \exists_x d$. Now, if P is able to make a step, thus reducing to P' and transforming the local store into e', what we see from the external point of view is that the process is transformed into $\exists_x^{e'} P'$, and that the information $\exists_x e$ present in the global store is transformed into $\exists_x e'$. To show how this works we show an instructive example.

Example 2. *Consider the constraint system from Example 1 and let* $Var = \{x\}$. *Let* $P = \exists_x^e(\mathbf{ask}\ (x > 10)\ \rightarrow\ Q)$ *with local store* $e = x > 42$*, and global store* $d = x > 2$.

$$\begin{array}{ll} \textit{R2} & \dfrac{(x > 10) \sqsubseteq e \sqcup \exists_x d = (x > 42 \sqcup \exists_x (x > 2))}{\langle \mathbf{ask}\ (x > 10)\ \rightarrow\ Q, e \sqcup \exists_x d\rangle \longrightarrow \langle Q, e \sqcup \exists_x d\rangle} \\ \textit{R5} & \dfrac{}{\langle P, d\rangle \longrightarrow \langle \exists_x^e Q, d \sqcup \exists_x e\rangle} \end{array}$$

Note that the x *in* d *is hidden, by using existential quantification in the reduction obtained by Rule R2. This expresses that the* x *in* d *is different from the*

one bound by the local process. Otherwise the ask process $\mathbf{ask}\ (x > 10)\ \rightarrow\ Q$ *would not be executed since the guard is not entailed by the global store* d*. Rule R2 applies since* $(x > 10) \sqsubseteq e \sqcup \exists_x d$*. Note that the free* x *in* $e \sqcup \exists_x d$ *is hidden in the global store, i.e.* $d \sqcup \exists_x e$*, to indicate that is different from the global* x*.*

Remark 3. *Observe that any transition is generated either by a process* $\mathbf{tell}(c)$ *or by a process* $\mathbf{ask}\ (c)\ \rightarrow\ Q$*. We say that a process* P *is active in a transition* $t = \gamma \longrightarrow \gamma'$ *if it generates such transition; i.e if there exist a derivation of* t *where R1 or R2 are used to produce a transition of the form* $\langle P, d\rangle \longrightarrow \gamma''$*.*

Barbed Semantics. The authors in [10] introduced a barbed semantics for CCP. Barbed equivalences have been introduced in [29] for CCS, and have become the standard behavioral equivalences for formalisms equipped with unlabeled reduction semantics. Intuitively, *barbs* are basic observations (predicates) on the states of a system. In the case of CCP, barbs are taken from the underlying set Con_0 of the constraint system.

Definition 2 (Barbs)**.** *A configuration* $\gamma = \langle P, d\rangle$ *is said to* satisfy *the barb* c *(*$\gamma \downarrow_c$*) iff* $c \sqsubseteq d$*. Similarly,* γ *satisfies a* weak barb c *(*$\gamma \Downarrow_c$*) iff there exist* γ' *s.t.* $\gamma \longrightarrow^* \gamma' \downarrow_c$*.*

To explain this notion consider the following example.

Example 3. *Consider the constraint system from Example 1 and let* $Var = \{x\}$*. Let* $\gamma = \langle \mathbf{ask}\ (x > 10)\ \rightarrow\ \mathbf{tell}(x > 42), x > 10\rangle$*. We have* $\gamma \downarrow_{x>5}$ *since* $(x > 5) \sqsubseteq (x > 10)$ *and* $\gamma \Downarrow_{x>42}$ *since* $\gamma \longrightarrow \langle \mathbf{tell}(x > 42), x > 10\rangle \longrightarrow \langle \mathbf{stop}, (x > 42)\rangle \downarrow_{x>42}$*.*

In this context, the equivalence proposed is the *saturated bisimilarity* [8, 9]. Intuitively, in order for two states to be saturated bisimilar, then *(i)* they should expose the same barbs, *(ii)* whenever one of them moves then the other should reply and arrive at an equivalent state (i.e. follow the bisimulation game), *(iii)* they should be equivalent under all the possible contexts of the language. In the case of CCP, it is enough to require that bisimulations are *upward closed* as in condition *(iii)* below.

Definition 3 (Saturated Barbed Bisimilarity)**.** *A saturated barbed bisimulation is a symmetric relation* $\mathcal{R}$ *on configurations s.t. whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$ *with* $\gamma_1 = \langle P, c\rangle$ *and* $\gamma_2 = \langle Q, d\rangle$ *implies that:*

(i) if $\gamma_1 \downarrow_e$ *then* $\gamma_2 \downarrow_e$*,*

(ii) if $\gamma_1 \longrightarrow \gamma_1'$ *then there exists* γ_2' *s.t.* $\gamma_2 \longrightarrow \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$*,*

(iii) for every $a \in Con_0$*,* $(\langle P, c \sqcup a\rangle, \langle Q, d \sqcup a\rangle) \in \mathcal{R}$*.*

We say that γ_1 *and* γ_2 *are saturated barbed bisimilar (*$\gamma_1 \dot{\sim}_{sb} \gamma_2$*) if there exists a saturated barbed bisimulation* $\mathcal{R}$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$*.*

We use the term "saturated" to be consistent with the original idea in [8, 9]. However, "saturated" in this context has nothing to do with the Milner's "saturation" for weak bisimilarity. In the following, we will continue to use "saturated" and "saturation" to denote these two different concepts.

Example 4. *Consider the constraint system from Example 1 and let* $Var = \{x\}$*. Take* $P = \mathbf{ask}\ (x > 5) \rightarrow \mathbf{stop}$ *and* $Q = \mathbf{ask}\ (x > 7) \rightarrow \mathbf{stop}$*. One can check that* $P \not\sim_{sb} Q$ *since* $\langle P, x > 5\rangle \longrightarrow$*, while* $\langle Q, x > 5\rangle \not\longrightarrow$*. Then consider* $\langle P + Q, true\rangle$ *and observe that* $\langle P + Q, true\rangle \dot{\sim}_{sb} \langle P, true\rangle$*. Indeed, for all constraints* e*, s.t.* $x > 5 \sqsubseteq e$*, both the configurations evolve into* $\langle \mathbf{stop}, e\rangle$*, while for all* e *s.t.* $x > 5 \not\sqsubseteq e$*, both configurations cannot proceed. Since* $x > 5 \sqsubseteq x > 7$*, the behavior of* Q *is somehow absorbed by the behavior of* P*.*

As we mentioned before, we are interested in deciding the weak version of the notion above. Then, *weak saturated barbed bisimilarity* ($\dot{\approx}_{sb}$) is obtained from Definition 3 by replacing the strong barbs in condition *(i)* for its weak version ($\Downarrow$) and the transitions in condition *(ii)* for the reflexive and transitive closure of the transition relation ($\longrightarrow^*$). Formally,

Definition 4 (Weak Saturated Barbed Bisimilarity)**.** *A weak saturated barbed bisimulation is a symmetric relation* $\mathcal{R}$ *on configurations s.t. whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$ *with* $\gamma_1 = \langle P, c\rangle$ *and* $\gamma_2 = \langle Q, d\rangle$ *implies that:*

(i) if $\gamma_1 \Downarrow_e$ *then* $\gamma_2 \Downarrow_e$*,*

(ii) if $\gamma_1 \longrightarrow^* \gamma_1'$ *then there exists* γ_2' *s.t.* $\gamma_2 \longrightarrow^* \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$*,*

(iii) for every $a \in Con_0$*,* $(\langle P, c \sqcup a\rangle, \langle Q, d \sqcup a\rangle) \in \mathcal{R}$*.*

We say that γ_1 *and* γ_2 *are weak saturated barbed bisimilar (*$\gamma_1 \dot{\approx}_{sb} \gamma_2$*) if there exists a weak saturated barbed bisimulation* $\mathcal{R}$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$*.*

To better understand this definition, consider the following example.

$$\text{LR1}\ \langle \mathbf{tell}(c), d\rangle \xrightarrow{true} \langle \mathbf{stop}, d \sqcup c\rangle \qquad \text{LR2}\ \frac{\alpha \in \min\{a \in Con_0 \mid c \sqsubseteq d \sqcup a\}}{\langle \mathbf{ask}\ (c)\ \rightarrow\ P, d\rangle \xrightarrow{\alpha} \langle P, d \sqcup \alpha\rangle}$$

$$\text{LR3}\ \frac{\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \xrightarrow{\alpha} \langle P' \parallel Q, d'\rangle} \qquad \text{LR4}\ \frac{\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle}{\langle P + Q, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle}$$

$$\text{LR5}\ \frac{\langle P[z/x], e[z/x] \sqcup d\rangle \xrightarrow{\alpha} \langle P', e' \sqcup d \sqcup \alpha\rangle}{\langle \exists^{e}_{x} P, d\rangle \xrightarrow{\alpha} \langle \exists^{e'[x/z]}_{x} P'[x/z], \exists_x(e'[x/z]) \sqcup d \sqcup \alpha\rangle}$$

$$\textit{with}\ x \notin fv(e'), z \notin fv(P) \cup fv(e \sqcup d \sqcup \alpha)$$

Table 3: Labeled semantics for CCP (symmetric rules for LR3 and LR4 are omitted).

Example 5. *Consider P and Q as in Example 4. We shall prove that $P \dot{\approx}_{sb} Q$. First notice that $\langle P, true\rangle \not\longrightarrow$ and also $\langle Q, true\rangle \not\longrightarrow$. Moreover, since none of the processes has the ability of adding information to the store, then for all $a \in Con_0$, $\langle P, a\rangle \Downarrow_e$ iff $e \sqsubseteq a$ iff $\langle Q, a\rangle \Downarrow_e$. Recall from Example 4 that $P \not\dot{\approx}_{sb} Q$ since an observer can plug P into $x > 5$ and observe a reduction, which cannot be observed with Q. Instead $P \dot{\approx}_{sb} Q$ since, intuitively, the discriminating power of $\dot{\approx}_{sb}$ cannot observe any reductions.*

2.2.4. Labeled Semantics

As explained in [10], in a transition of the form $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle$ the label α represents a *minimal* information (from the environment) that needs to be added to the store d to evolve from $\langle P, d\rangle$ into $\langle P', d'\rangle$, i.e., $\langle P, d \sqcup \alpha\rangle \longrightarrow \langle P', d'\rangle$. As a consequence, the transitions labeled with the constraint $true$ are in one to one correspondence with the reductions defined in the previous section. For this reason, hereafter we will sometimes write $\longrightarrow$ to mean $\xrightarrow{true}$. Before formally introducing the labeled semantics, we fix some notation.

Notation 1. *We will use $\rightsquigarrow$ to denote a generic transition relation on the state space $Conf$ and labels Con_0. Also in this case $\rightsquigarrow$ mean $\overset{true}{\rightsquigarrow}$. Also, given a set of*

initial configurations IS, we shall use $\texttt{Config}_{\rightsquigarrow}(IS)$ *to denote the set of reachable configurations*[3] *using* $\rightsquigarrow$*, i.e.:*

$$\texttt{Config}_{\rightsquigarrow}(IS) = \{\gamma' \mid \exists \gamma \in IS \textit{ s.t. } \gamma \stackrel{\alpha_1}{\rightsquigarrow} \ldots \stackrel{\alpha_n}{\rightsquigarrow} \gamma' \textit{ for some } n \geq 0\}$$

The LTS $(Conf, Con_0, \longrightarrow)$ is defined by the rules in Table 3. The rule LR2, for example, says that $\langle \mathbf{ask}\ (c)\ \rightarrow\ P, d\rangle$ can evolve to $\langle P, d \sqcup \alpha\rangle$ if the environment provides a minimal constraint α that added to the store d entails c, i.e., $\alpha \in \min\{a \in Con_0 \mid c \sqsubseteq d \sqcup a\}$. Note that assuming that $(Con, \sqsubseteq)$ is well-founded (Remark 1) is necessary to guarantee that α exists whenever $\{a \in Con_0 \mid c \sqsubseteq d \sqcup a\}$ is not empty. The rule LR5 in Table 3 uses in the antecedent derivation a fresh variable z that acts as a substitute for the free occurrences of x in P and its local store e. (Recall that $T[z/x]$ represents T with x replaced with z). This way we identify with z the free occurrences of x in P and e and avoid clashes with those in α and d. The other rules are easily seen to realize the intuition given in Section 2.2.2. Figure 2 illustrates the LTSs of our running example.

To better understand the labeled semantics consider the following example.

Example 6. *Let* $\gamma_1 = \langle \mathbf{tell}(true), true\rangle$ *and* $\gamma_2 = \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), true\rangle$. *We can show that* $\gamma_1 \dot{\approx}_{sb} \gamma_2$ *when* $d \sqsubseteq c$. *Intuitively, this corresponds to the fact that the implication* $c \Rightarrow d$ *is equivalent to* $true$ *when* c *entails* d. *Note that* $LTS_{\longrightarrow}(\gamma_1)$ *and* $LTS_{\longrightarrow}(\gamma_2)$ *are the following:* $\gamma_1 \longrightarrow \langle \mathbf{stop}, true\rangle$ *and* $\gamma_2 \stackrel{c}{\longrightarrow} \langle \mathbf{tell}(d), c\rangle \longrightarrow \langle \mathbf{stop}, c\rangle$. *Consider now the relation* $\mathcal{R}$ *as follows:*

$$\mathcal{R} = \{(\gamma_2, \gamma_1),\ (\gamma_2, \langle \mathbf{stop}, true\rangle),\ (\langle \mathbf{tell}(d), c\rangle, \langle \mathbf{stop}, c\rangle),\ (\langle \mathbf{stop}, c\rangle, \langle \mathbf{stop}, c\rangle)\}$$

One can verify that the symmetric closure of $\mathcal{R}$ *is a weak saturated barbed bisimulation as in Definition 3.*

The labeled semantics is *sound* and *complete* w.r.t. the unlabeled one. Soundness states that $\langle P, d\rangle \stackrel{\alpha}{\longrightarrow} \langle P', d'\rangle$ corresponds to our intuition that if α is added to d, P can reach $\langle P', d'\rangle$. Completeness states that if we add a to (the store in) $\langle P, d\rangle$ and reduce to $\langle P', d'\rangle$, it exists a minimal information $\alpha \sqsubseteq a$ such that $\langle P, d\rangle \stackrel{\alpha}{\longrightarrow} \langle P', d''\rangle$ with $d'' \sqsubseteq d'$.

The following lemma is an extension of the one in [10] which considers nondeterministic CCP.

[3]Notice that when $IS^{\star}_{\rightsquigarrow}$ denotes the set of reachable configurations then $\texttt{Config}_{\rightsquigarrow}(IS) = IS^{\star}_{\rightsquigarrow}$.

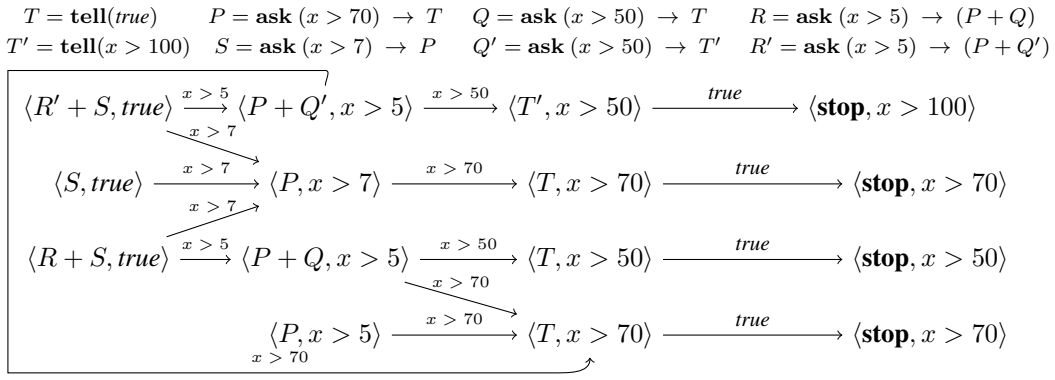

Figure 2: $LTS_{\longrightarrow}(\{\langle R' + S, true\rangle, \langle S, true\rangle, \langle R + S, true\rangle, \langle P, x > 5\rangle\})$.

Lemma 1 (Correctness of $\longrightarrow$)**.**
(Soundness) If $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ *then* $\langle P, c \sqcup \alpha\rangle \longrightarrow \langle P', c'\rangle$.
(Completeness) If $\langle P, c \sqcup a\rangle \longrightarrow \langle P', c'\rangle$ *then there exist* α *and* b *such that* $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c''\rangle$ *where* $\alpha \sqcup b = a$ *and* $c'' \sqcup b = c'$.

Proof. (Soundness) By induction on (the depth) of the inference of $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle$. Here we consider only the additional case for the non-deterministic choice.

- Using Rule LR4 (Table 3), take $P = Q + R$ and $P' = Q'$ then we have $\langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d'\rangle$ by a shorter inference. By appeal to induction then $\langle Q, d \sqcup \alpha\rangle \longrightarrow \langle Q', d'\rangle$. Applying Rule R4 (Table 2) to the previous reduction we get $\langle Q + R, d \sqcup \alpha\rangle \longrightarrow \langle Q', d'\rangle$.

(Completeness) The proof proceeds by induction on (the depth) of the inference of $\langle P, d \sqcup a\rangle \longrightarrow \langle P', d'\rangle$. Again, we consider only the case for the non-deterministic choice.

- Using Rule R4 (Table 2), take $P = Q + R$ and $P' = Q'$ then we have $\langle Q, d \sqcup a\rangle \longrightarrow \langle Q', d'\rangle$ by a shorter inference. Note that the active process generating the transition could be either an ask or a tell (See Remark 3). Let suppose that the constraint that has been either asked or told is c. If it is generated by an ask then $d' = d \sqcup a$ and $c \sqsubseteq d \sqcup a$. Note that $a \in \{a' \in Con_0 | c \sqsubseteq d \sqcup a'\}$, and then by Remark 1 there exists $\alpha \in min(\{a' \in Con_0 | c \sqsubseteq d \sqcup a'\})$ such that $\alpha \sqsubseteq a$. If it is generated by a tell then $d' = d \sqcup a \sqcup c$, thus in both cases is safe to assume that $d' = d \sqcup a \sqcup c$. Thereafter

by induction hypothesis we have that there exist α and b such that:

$$\langle Q, d\rangle \xrightarrow{\alpha} \langle Q', d''\rangle$$

with $a = \alpha \sqcup b$ and $d' = d'' \sqcup b$. As said formerly the active process generating this transition could be either an ask or a tell. If it is generated by an ask then $d'' = d \sqcup \alpha$ and if it is generated by a tell then $\alpha = true$ and $d'' = d \sqcup c$. Hence in both cases is safe to assume that $d'' = d \sqcup c$. Now by Rule LR4 we have that

$$\langle Q + R, d\rangle \xrightarrow{\alpha} \langle Q', d''\rangle$$

since $d'' = d \sqcup c$ we say that $a = b$ and that $d' = d \sqcup a \sqcup c = d \sqcup b \sqcup c = d'' \sqcup b$.

□

The above lemma is central for deciding bisimilarity in CCP. In fact, we will show later that, for the weak (saturated) semantics, the completeness direction does not hold. From this we will show that the standard reduction from weak to strong does not work.

2.3. Deciding strong bisimilarity for CCP

In this section we recall how to check $\dot{\sim}_{sb}$ with a modified version of partition refinement introduced in [12]. Henceforth, we shall refer to this algorithm as *CCP partition refinement*.

2.3.1. Equivalences: Saturated Barbed, Irredundant and Symbolic Bisimilarity

The main problem with checking $\dot{\sim}_{sb}$ is the quantification over all contexts. This problem is addressed in [12] following the abstract approach in [30]. More precisely, we use an equivalent notion, namely *irredundant bisimilarity* $\dot{\sim}_I$, which can be verified with CCP partition refinement. As its name suggests, $\dot{\sim}_I$ only takes into account those transitions deemed irredundant. However, technically speaking, going from $\dot{\sim}_{sb}$ to $\dot{\sim}_I$ requires one intermediate notion, so-called *symbolic bisimilarity*. These three notions are shown to be equivalent, $\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_I$. In the following we recall all of them.

Let us first give some auxiliary definitions. Unlike for the standard partition refinement, we need to consider a particular kind of transitions, so-called *irredundant transitions*. These are those transitions that are not dominated by others, in a given partition, in the sense defined below.

Definition 5 (Transition Derivation)**.** *Let t and t' be two transitions of the form $t = (\gamma, \alpha, \langle P', c' \rangle)$ and $t' = (\gamma, \beta, \langle P', c'' \rangle)$. We say that t derives t', written $t \vdash_D t'$, iff $\alpha \sqsubseteq \beta$ and $c'' = c' \sqcup \beta$.*

The intuition is that the transition t dominates t' iff t requires less (or equal) information from the environment than t' does (hence $\alpha \sqsubseteq \beta$), and they end up in configurations which differ only by the additional information in β not present in α (hence $c'' = c' \sqcup \beta$). To better explain this notion let us give an example.

Example 7. *Consider the following process:*

$$P = (\mathbf{ask}\ (x > 10)\ \rightarrow\ \mathbf{tell}(x > 42)) + (\mathbf{ask}\ (x > 15)\ \rightarrow\ \mathbf{tell}(x > 42))$$

and let $\gamma = \langle P, true \rangle$. Now let t_1 and t_2 be transitions defined as:

$$t_1 = \gamma \xrightarrow{x>10} \langle \mathbf{tell}(x > 42), x > 10 \rangle \text{ and } t_2 = \gamma \xrightarrow{x>15} \langle \mathbf{tell}(x > 42), x > 15 \rangle$$

One can check that $t_1 \vdash_D t_2$ since $(x > 10) \sqsubseteq (x > 15)$ and also $(x > 15) = ((x > 15) \sqcup (x > 10))$.

Notice that in the definition above t and t' end up in configurations whose processes are *syntactically identical* (i.e., P'). The following notion parameterizes the notion of dominance w.r.t. a relation on configurations $\mathcal{R}$ (rather than fixing it to the identity on configurations).

Definition 6 (Transition Derivation w.r.t. $\mathcal{R}$)**.** *We say that the transition t derives a transition t' w.r.t a relation on configurations $\mathcal{R}$, written $t \vdash_{\mathcal{R}} t'$, iff there exists t'' such that $t \vdash_D t''$, $lab(t'') = lab(t')$ and $tar(t'') \mathcal{R}\ tar(t')$.*[4]

To better understand this definition consider the following example.

Example 8. *Consider the following processes:*

$$Q_1 = (\mathbf{ask}\ (b)\ \rightarrow\ (\mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d))) \text{ and } Q_2 = (\mathbf{ask}\ (a)\ \rightarrow\ \mathbf{stop})$$

Now let $P = Q_1 + Q_2$ where $d \sqsubseteq c$ and $a \sqsubset b$. Then take $\gamma = \langle P, true \rangle$ and consider the transitions t and t' as:

$$t = \gamma \xrightarrow{a} \langle \mathbf{stop}, a \rangle \text{ and } t' = \gamma \xrightarrow{b} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), b \rangle$$

[4]Recall that for a given transition $t = (s, a, r)$ the source, the target and the label are $src(t) = s$, $tar(t) = r$ and $lab(t) = a$ respectively.

Finally, let $\mathcal{R} = \dot{\approx}_{sb}$ *and take* $t'' = (\gamma, b, \langle \mathbf{stop}, b\rangle)$ *which is constructed using the target process in* t *and the label in* t'*. One can check that* $t \vdash_{\mathcal{R}} t'$ *as in Definition 8. Firstly,* $t \vdash_D t''$ *follows from* $a \sqsubset b$*. Secondly, we know* $tar(t'')\mathcal{R}\ tar(t')$ *from Example 6, i.e.* $\langle \mathbf{stop}, b\rangle \dot{\approx}_{sb} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), b\rangle$ *since* $\langle \mathbf{stop}, true\rangle \dot{\approx}_{sb} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), true\rangle$.

Now we introduce the concept of *domination*, which consists in strengthening the notion of derivation by requiring labels to be different.

Definition 7 (Transition Domination)**.** *Let* t *and* t' *be two transitions of the form* $t = (\gamma, \alpha, \langle P', c'\rangle)$ *and* $t' = (\gamma, \beta, \langle P', c''\rangle)$*. We say that* t *dominates* t'*, written* $t \succ_D t'$*, iff* $t \vdash_D t'$ *and* $\alpha \neq \beta$*.*

As we did for derivation, we now parameterize the notion of domination as follows.

Definition 8 (Transition Domination w.r.t. $\mathcal{R}$ and Irredundant Transition w.r.t. $\mathcal{R}$)**.** *We say that the transition* t *dominates a transition* t' *w.r.t a relation on configurations* $\mathcal{R}$*, written* $t \succ_{\mathcal{R}} t'$*, iff there exists* t'' *such that* $t \succ_D t''$*,* $lab(t'') = lab(t')$ *and* $tar(t'')\mathcal{R}\ tar(t')$*. A transition is said to be redundant w.r.t. to* $\mathcal{R}$ *when it is dominated by another w.r.t.* $\mathcal{R}$*, otherwise it is said to be irredundant w.r.t. to* $\mathcal{R}$*.*

We are now able to introduce symbolic bisimilarity. Intuitively, two configurations γ_1 and γ_2 are symbolic bisimilar iff (i) they have the same barbs and (ii) whenever there is a transition from $\gamma_1 \xrightarrow{\alpha} \gamma_1'$, then we require that γ_2 must reply with a similar transition $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ (where γ_1' and γ_2' are now equivalent) or some other transition that derives it. In other words, the move from the defender does not need to use exactly the same label, but a transition that is "stronger" (in terms of derivation $\vdash_D$) could also do the job. Formally we have the definition below.

Definition 9 (Symbolic Bisimilarity)**.** *A symbolic bisimulation is a symmetric relation* $\mathcal{R}$ *on configurations s.t. whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$ *with* $\gamma_1 = \langle P, c\rangle$ *and* $\gamma_2 = \langle Q, d\rangle$ *implies that:*

(i) if $\gamma_1 \downarrow_e$ *then* $\gamma_2 \downarrow_e$*,*

(ii) if $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ *then there exists a transition* $t = \langle Q, d\rangle \xrightarrow{\beta} \langle Q', d''\rangle$ *and a store* d' *s.t.* $t \vdash_D \langle Q, d\rangle \overset{\alpha}{\rightsquigarrow} \langle Q', d'\rangle$ *and* $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$*.*

We say that γ_1 *and* γ_2 *are symbolic bisimilar (*$\gamma_1 \dot{\sim}_{sym} \gamma_2$*) if there exists a symbolic bisimulation* $\mathcal{R}$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$*.*

Let us illustrate the definition above by using the process in Figure 2.

Example 9. *Consider the following processes:*

$$T = \mathbf{tell}(true), P = \mathbf{ask}\ (x > 5)\ \rightarrow\ T \text{ and } Q = \mathbf{ask}\ (x > 7)\ \rightarrow\ T$$

We provide a symbolic bisimulation

$$\mathcal{R} = \{(\langle P + Q, true\rangle, \langle P, true\rangle)\}\ \cup\ id$$

to prove $\langle P + Q, true\rangle \dot{\sim}_{sym} \langle P, true\rangle$. *Take the pair* $(\langle P + Q, true\rangle, \langle P, true\rangle)$. *The first condition in Definition 9 is trivial. As for the second one, let us take the transition* $\langle P + Q, true\rangle \xrightarrow{x>7} \langle T, x > 7\rangle$. *We have to check that* $\langle P, true\rangle$ *is able to reply with a stronger transition. Thus consider the transitions* t *and* t' *as follows:*

$$t = \langle P, true\rangle \xrightarrow{x>5} \langle T, x > 5\rangle \text{ and } t' = \langle P, true\rangle \overset{x>7}{\rightsquigarrow} \langle T, x > 7\rangle$$

Then we can observe that $t \vdash_D t'$ *and* $\langle T, x > 7\rangle \mathcal{R} \langle T, x > 7\rangle$. *The remaining pairs are trivially verified.*

And finally, the irredundant version, which follows the standard bisimulation game where labels need to be matched, however only those so-called irredundant transitions must be considered.

Definition 10 (Irredundant Bisimilarity)**.** *An irredundant bisimulation is a symmetric relation* $\mathcal{R}$ *on configurations s.t. whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$ *implies that:*

(i) if $\gamma_1 \downarrow_e$ *then* $\gamma_2 \downarrow_e$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ *and it is irredundant in* $\mathcal{R}$ *then there exists* γ_2' *s.t.* $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 *and* γ_2 *are irredundant bisimilar* ($\gamma_1\ \dot{\sim}_I\ \gamma_2$) *if there exists an irredundant bisimulation* $\mathcal{R}$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Example 10. *Let* T, P *and* Q *as in Example 9. We can verify that the relation* $\mathcal{R}$ *in Example 9 is an irredundant bisimulation to show that* $\langle P + Q, true\rangle \dot{\sim}_I \langle P, true\rangle$. *We take the pair* $(\langle P + Q, true\rangle, \langle P, true\rangle)$. *The first item in Definition 10 is obvious. Notice now that* $\langle P + Q, true\rangle \xrightarrow{x>5} \langle T, x > 5\rangle$ *is irredundant (Definition 8), thus this transition must be matched by* $\langle P, true\rangle$. *Hence we can take*

$\langle P, true\rangle \xrightarrow{x>5} \langle T, x > 5\rangle$ *and then* $\langle T, x > 5\rangle \mathcal{R} \langle T, x > 5\rangle$. *The other pairs are trivially verified. Notice that*

$$\langle P + Q, true\rangle \xrightarrow{x>5} \langle T, x > 5\rangle \succ_{\mathcal{R}} \langle P + Q, true\rangle \xrightarrow{x>7} \langle T, x > 7\rangle$$

hence $\langle P + Q, true\rangle \xrightarrow{x>7} \langle T, x > 7\rangle$ *is redundant since* $(x > 5) \sqsubset (x > 7)$, *therefore it does not need to be matched by* $\langle P, true\rangle$.

As we said at the beginning, the above-defined equivalences coincide with saturated barbed bisimilarity ($\dot{\sim}_{sb}$, Definition 3). The proof, given in [12], strongly relies on Lemma 1.

Theorem 1 ([12]). $\langle P, c\rangle \dot{\sim}_I \langle Q, d\rangle$ *iff* $\langle P, c\rangle \dot{\sim}_{sym} \langle Q, d\rangle$ *iff* $\langle P, c\rangle \dot{\sim}_{sb} \langle Q, d\rangle$

Using this theorem we can define a modified partition refinement, that consider only the irredundant transitions, in order to compute $\dot{\sim}_{sb}$ as we shall see in the next section.

2.3.2. Partition Refinement for CCP

In [12] we adapted the standard partition refinement procedure to decide strong bisimilarity for CCP ($\dot{\sim}_{sb}$). As we did for the standard partition refinement, we start with $\texttt{Config}_{\longrightarrow}(IS)$, that is the set of all states that are reachable from the set of initial state IS using $\longrightarrow$. However, in the case of CCP some other states must be added to $IS^{\star}_{\rightsquigarrow}$ in order to verify $\dot{\sim}_{sb}$ as we will explain later on.

First, since configurations satisfying different barbs are surely different, the algorithm can be safely started with a partition that equates each state satisfying the same barbs. Hence, as initial partition of $IS^{\star}_{\rightsquigarrow}$, we take $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$, where γ and γ' are in B_i iff they satisfy the same barbs.

We now explain briefly how to compute $IS^{\star}_{\rightsquigarrow}$ using the Rules in Table 4. Rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$ and $(\mathsf{RS}^{IS}_{\rightsquigarrow})$ say that all the reachable states using the transition relation $\rightsquigarrow$ from the set of initial states should be included, i.e., $\texttt{Config}_{\rightsquigarrow}(IS) \subseteq IS^{\star}_{\rightsquigarrow}$.

The rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ adds the additional states needed to check redundancy. Consider the transitions $t_1 = \gamma \overset{\alpha}{\rightsquigarrow} \langle P_1, c_1\rangle$ and $t_2 = \gamma \overset{\beta}{\rightsquigarrow} \langle P_2, c_2\rangle$ with $\alpha \sqsubset \beta$ and $c_2 = c_1 \sqcup \beta$ in Rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$. Suppose that at some iteration of the partition refinement algorithm the current partition is $\mathcal{P}$ and that $\langle P_2, c_2\rangle \mathcal{P} \langle P_1, c_2\rangle$. Then, according to Definition 8 the transitions t_1 would dominate t_2 w.r.t $\mathcal{P}$. This makes t_2 redundant w.r.t $\mathcal{P}$. Since $\langle P_1, c_2\rangle$ may allow us to witness a potential redundancy of t_2, we include it in $IS^{\star}_{\rightsquigarrow}$ (and thus, from the definition of the initial

$$(\mathsf{IS}^{IS}_{\rightsquigarrow})\ \frac{\gamma \in IS}{\gamma \in IS^{\star}_{\rightsquigarrow}} \qquad (\mathsf{RS}^{IS}_{\rightsquigarrow})\ \frac{\gamma \in IS^{\star}_{\rightsquigarrow} \quad \gamma \overset{\alpha}{\rightsquigarrow} \gamma'}{\gamma' \in IS^{\star}_{\rightsquigarrow}}$$

$$(\mathsf{RD}^{IS}_{\rightsquigarrow})\ \frac{\gamma \in IS^{\star}_{\rightsquigarrow} \quad t_1 = \gamma \overset{\alpha}{\rightsquigarrow} \langle P_1, c_1\rangle \quad t_2 = \gamma \overset{\beta}{\rightsquigarrow} \langle P_2, c_2\rangle \quad \alpha \sqsubset \beta \quad c_2 = c_1 \sqcup \beta}{\langle P_1, c_2\rangle \in IS^{\star}_{\rightsquigarrow}}$$

Table 4: Rules for generating the states used in the partition refinement for ccp

partition $\mathcal{P}^0$, also in the block of $\mathcal{P}^0$ where $\langle P_2, c_2\rangle$ is). See [12] for further details about the computation of $IS^{\star}_{\rightsquigarrow}$.

Finally, instead of using the function $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$ of Algorithm 1, the partitions are refined by employing the function $\mathbf{IR}_{\rightsquigarrow}(\mathcal{P})$ defined as follows:

Definition 11. *(Refinement function for CCP) Given a partition $\mathcal{P}$ we define $\mathbf{IR}_{\rightsquigarrow}(\mathcal{P})$ as follows: $\gamma_1\ \mathbf{IR}_{\rightsquigarrow}(\mathcal{P})\ \gamma_2$ iff*

if $\gamma_1 \overset{\alpha}{\rightsquigarrow} \gamma_1'$ is irredundant w.r.t. $\mathcal{P}$ then there exists γ_2' s.t. $\gamma_2 \overset{\alpha}{\rightsquigarrow} \gamma_2'$ and $\gamma_1'\,\mathcal{P}\,\gamma_2'$

Using this function, the partition refinement algorithm for CCP is defined as follows:

Algorithm 2 `pr-ccp`$(IS, \rightsquigarrow)$

Initialization

1. Compute $IS^{\star}_{\rightsquigarrow}$ with the rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ defined in Table 4,
2. $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$ is a partition of $IS^{\star}_{\rightsquigarrow}$ where γ and γ' are in B_i iff they satisfy the same barbs ($\downarrow_c$),

Iteration $\mathcal{P}^{n+1} := \mathbf{IR}_{\rightsquigarrow}(\mathcal{P}^n)$ as in Definition 11
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

Algorithm 2 can be used to decide $\dot{\sim}_{sb}$ with exponential time whenever the set of reachable states is finite. (Recall that $\mathtt{Config}_{\longrightarrow}(IS)$ represents the set of states that are reachable from the initial states IS using $\longrightarrow$.) More precisely:

Theorem 2. *([12]) Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of $\mathtt{pr\text{-}ccp}(IS, \longrightarrow)$ in Algorithm 2. If $\mathtt{Config}_{\longrightarrow}(IS)$ is finite then:*

- $\gamma \; \mathcal{P} \; \gamma'$ *iff* $\gamma \; \dot{\sim}_{sb} \; \gamma'$.
- $\mathtt{pr\text{-}ccp}(IS, \longrightarrow)$ *may take exponential time in the size of* $\mathtt{Config}_{\longrightarrow}(IS)$.

The exponential time is due to construction of the set $IS^{\star}_{\longrightarrow}$ (Algorithm 2, step 1) whose size is exponential in $|\mathtt{Config}_{\longrightarrow}(IS)|$.

2.4. Incompleteness of Milner's saturation method in CCP

As mentioned at the beginning of this section, the standard approach for deciding weak equivalences is to add some transitions to the original processes, so-called *saturation*, and then check for the strong equivalence. In calculi like CCS, such saturation consists in forgetting about the internal actions that make part of a sequence containing one observable action (Table 5). However, for CCP this method does not work. The problem is that the transition relation proposed by Milner is not complete for CCP, hence the relation among the saturated, symbolic and irredundant equivalences is broken. In the next section we will provide a stronger saturation, which is complete, and allows us to use the CCP partition refinement to compute $\dot{\approx}_{sb}$.

Let us show why Milner's approach does not work. First, we need to introduce formally the concept of *completeness* for a given transition relation.

Definition 12 (Completeness)**.** *We say that $\rightsquigarrow$ is complete if and only if whenever $\langle P, c \sqcup a\rangle \rightsquigarrow \langle P', c'\rangle$ then there exist $\alpha, b \in Con_0$ s.t. $\langle P, c\rangle \stackrel{\alpha}{\rightsquigarrow} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$.*

Notice that $\longrightarrow$ (i.e the reduction semantics, see Table 2) is complete, and it corresponds to the second item of Lemma 1. Now Milner's method defines a

MR1	$\dfrac{\gamma \stackrel{\alpha}{\longrightarrow} \gamma'}{\gamma \stackrel{\alpha}{\Longrightarrow} \gamma'}$	MR2 $\quad \dfrac{}{\gamma \Longrightarrow \gamma}$
	MR3 $\quad \dfrac{\gamma \Longrightarrow \gamma_1 \stackrel{\alpha}{\Longrightarrow} \gamma_2 \Longrightarrow \gamma'}{\gamma \stackrel{\alpha}{\Longrightarrow} \gamma'}$	

Table 5: Milner's Saturation Method for CCP

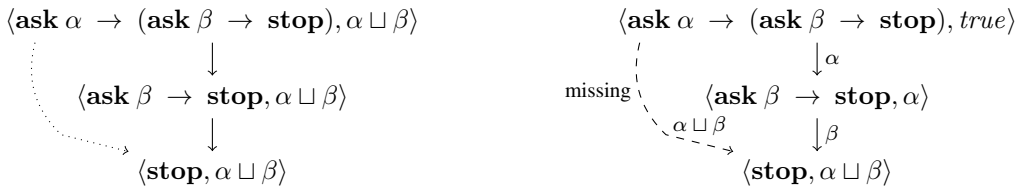

Figure 3: Counterexample for completeness using Milner's saturation method (cycles from MR2 omitted). Both graphs are obtained using Table 5.

new transition relation $\Longrightarrow$ using the rules in Table 5, but it turns out not to be complete.

Proposition 1. *The relation $\Longrightarrow$ defined in Table 5 is not complete.*

Proof. We will show a counter-example where the completeness for $\Longrightarrow$ does not hold. Let $P = \mathbf{ask}\ \alpha\ \rightarrow\ (\mathbf{ask}\ \beta\ \rightarrow\ \mathbf{stop})$ and $d = \alpha \sqcup \beta$. Now consider the transition $\langle P, d\rangle \Longrightarrow \langle \mathbf{stop}, d\rangle$ and let us apply the completeness lemma, we can take $c = true$ and $a = \alpha \sqcup \beta$, therefore by completeness there must exist b and λ s.t. $\langle P, true\rangle \stackrel{\lambda}{\Longrightarrow} \langle \mathbf{stop}, c''\rangle$ where $\lambda \sqcup b = \alpha \sqcup \beta$ and $c'' \sqcup b = d$. However, notice that the only transition possible is $\langle P, true\rangle \stackrel{\alpha}{\Longrightarrow} \langle(\mathbf{ask}\ \beta\ \rightarrow\ \mathbf{stop}), \alpha\rangle$, hence completeness does not hold since there is no transition from $\langle P, true\rangle$ to $\langle \mathbf{stop}, c''\rangle$ for some c''. Figure 3 illustrates the problem. □

We can now use this fact to see why the method does not work for computing $\dot{\approx}_{sb}$ using CCP partition refinement. First, let us redefine some concepts using the new transition relation $\Longrightarrow$. Because of condition (i) in $\dot{\approx}_{sb}$, we need a new definition of barbs, namely *weak barbs w.r.t. $\Longrightarrow$*.

Definition 13. *We say γ satisfies a weak barb e w.r.t. $\Longrightarrow$, written $\gamma \Downarrow_e$, if and only if $\gamma \Longrightarrow^* \gamma' \downarrow_e$.*

We shall see later on that $\Downarrow$ coincides with $\Downarrow$. Using this notion, we introduce symbolic and irredundant bisimilarity w.r.t. $\Longrightarrow$, denoted by $\dot{\sim}^{\Longrightarrow}_{sym}$ and $\dot{\sim}^{\Longrightarrow}_{I}$ respectively. They are defined as in Definition 9 and 10 where in condition (i) weak barbs ($\Downarrow$) are replaced with $\Downarrow$ and in condition (ii) the transition relation is now $\Longrightarrow$. More precisely:

Definition 14 (Symbolic Bisimilarity over $\Longrightarrow$)**.** *A symbolic bisimulation over $\Longrightarrow$ is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c\rangle$ and $\gamma_2 = \langle Q, d\rangle$:*

(i) if $\gamma_1 \not\downarrow_e$ *then* $\gamma_2 \not\downarrow_e$,

(ii) if $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c'\rangle$ *then there exists a transition* $t = \langle Q, d\rangle \stackrel{\beta}{\Longrightarrow} \langle Q', d''\rangle$ *s.t.* $t \vdash_D \langle Q, d\rangle \stackrel{\alpha}{\rightsquigarrow} \langle Q', d'\rangle$ *and* $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$

We say that γ_1 *and* γ_2 *are symbolic bisimilar over* $\Longrightarrow$*, written* $\gamma_1 \dot{\sim}^{\Longrightarrow}_{sym} \gamma_2$*, if there exists a symbolic bisimulation over* $\Longrightarrow$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Definition 15 (Irredundant Bisimilarity over $\Longrightarrow$)**.** *An irredundant bisimulation over* $\Longrightarrow$ *is a symmetric relation* $\mathcal{R}$ *on configurations s.t. whenever* $(\gamma_1, \gamma_2) \in \mathcal{R}$*:*

(i) if $\gamma_1 \not\downarrow_e$ *then* $\gamma_2 \not\downarrow_e$,

(ii) if $\gamma_1 \stackrel{\alpha}{\Longrightarrow} \gamma_1'$ *and it is irredundant in* $\mathcal{R}$ *then there exists* γ_2' *s.t.* $\gamma_2 \stackrel{\alpha}{\Longrightarrow} \gamma_2'$ *and* $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 *and* γ_2 *are irredundant bisimilar over* $\Longrightarrow$*, written* $\gamma_1 \dot{\sim}^{\Longrightarrow}_{I} \gamma_2$*, if there exists an irredundant bisimulation over* $\Longrightarrow$ *s.t.* $(\gamma_1, \gamma_2) \in \mathcal{R}$.

One would expect that since $\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_I$ then it is the case that $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$, given that these new notions are supposed to be the weak versions of the former ones when using the saturation method. However, completeness is necessary for proving $\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_I$, and from Proposition 1 we know that $\Longrightarrow$ is not complete hence we might expect that $\dot{\approx}_{sb}$ does not imply $\dot{\sim}^{\Longrightarrow}_{sym}$ nor $\dot{\sim}^{\Longrightarrow}_{I}$. In fact, the following counter-example proves this.

Example 11. *Let* P, P' *and* Q *as in Figure 4 and 5. Let* $IS = \{\langle P, true\rangle, \langle Q, true\rangle\}$ *Figure 4 shows* $LTS_{\longrightarrow}(IS)$*. Figure 5 presents* $LTS_{\Longrightarrow}(IS)$ *where* $\Longrightarrow$ *is defined in Table 5 (Milner's saturation method).*

First, notice that $\langle P, true\rangle \dot{\approx}_{sb} \langle Q, true\rangle$*, since there exists a saturated weak barbed bisimulation:*

$$\mathcal{R} = \{(\langle P, true\rangle, \langle Q, true\rangle)\} \cup id$$

However, $\langle P, true\rangle \not\dot{\sim}^{\Longrightarrow}_{I} \langle Q, true\rangle$*. To prove that, we need to pick an irredundant transition from* $\langle P, true\rangle$ *or* $\langle Q, true\rangle$ *(after saturation) s.t. the other cannot match. Thus, take* $\langle Q, true\rangle \stackrel{\alpha \sqcup \beta}{\longrightarrow} \langle \mathbf{tell}(c), \alpha \sqcup \beta\rangle$ *which is irredundant and given that* $\langle P, true\rangle$ *does not have a transition labeled with* $\alpha \sqcup \beta$ *then we know that we cannot find an irredundant bisimulation containing* $(\langle P, true\rangle, \langle Q, true\rangle)$ *therefore* $\langle P, true\rangle \not\dot{\sim}^{\Longrightarrow}_{I} \langle Q, true\rangle$*. Using the same reasoning we can also show that* $\dot{\approx}_{sb}$ *does not imply* $\dot{\sim}^{\Longrightarrow}_{sym}$.

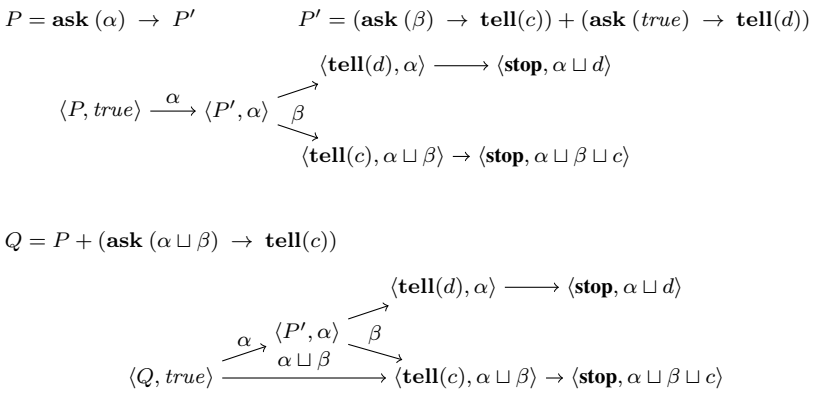

Figure 4: $LTS_{\longrightarrow}(\{\langle P, true\rangle, \langle Q, true\rangle\})$

3. Reducing weak bisimilarity to Strong in CCP

In this section we shall provide a method for deciding weak bisimilarity in CCP. As shown in Section 2.4, the usual method for deciding weak bisimilarity (introduced in Section 2.1) does not work for CCP. We shall proceed by redefining $\Longrightarrow$ in such a way that it is sound and complete for CCP. Then we prove that, w.r.t. $\Longrightarrow$, symbolic and irredundant bisimilarity coincide with $\dot{\approx}_{sb}$, i.e. $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$. We therefore conclude that the partition refinement algorithm in [12] can be used to verify $\dot{\approx}_{sb}$ w.r.t. $\Longrightarrow$.

3.1. Defining a new saturation method for CCP

If we analyze the counter-example to completeness (see Figure 3), one can see that the problem arises because of the nature of the labels in CCP, namely using this method $\langle \mathbf{ask}\ \alpha \;\rightarrow\; (\mathbf{ask}\ \beta \;\rightarrow\; \mathbf{stop}), true\rangle$ does not have a transition with $\alpha \sqcup \beta$ to $\langle \mathbf{stop}, \alpha \sqcup \beta\rangle$, hence that fact can be exploited to break the relation among the weak equivalences. Following this reasoning, instead of only forgetting about the silent actions we also take into account that labels in CCP can be added together. Thus we have a new rule that creates a new transition for each two consecutive ones, whose label is the lub of the labels in them. This method can also be thought as the reflexive and transitive closure of the labeled transition relation $\xrightarrow{\alpha}$. Such transition relation turns out to be sound and complete and it can be used to decide $\dot{\approx}_{sb}$.

The remaining part of this section is structured as follows. First we define a new saturation method and we proceed to prove that the weak barbs resulting

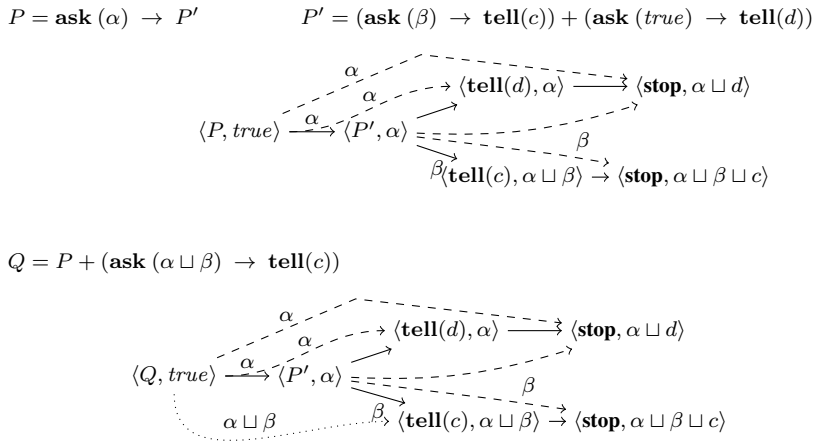

Figure 5: $LTS_{\Longrightarrow}(\{\langle P, true\rangle, \langle Q, true\rangle\})$ where $\Longrightarrow$ is defined in Table 5 (Milner's saturation method). The cycles from rule MR2 are omitted. The dashed transitions are those added by the rules in Table 5. The dotted transition is the (irredundant) one that $\langle Q, true\rangle$ can take but $\langle P, true\rangle$ cannot match, therefore showing that $\langle P, true\rangle \dot{\not\sim}_I^{\Longrightarrow} \langle Q, true\rangle$

from such method are consistent with that of CCP (as in Definition 2). Then under the assumption that $\longrightarrow$ is finitely branching we prove that $\Longrightarrow$ is also finitely branching. Moreover, we follow by showing how this method would be inaccurate in CCS-like formalisms since it could turn a finitely branching LTS into be infinitely branching one. With these elements we can prove soundness and completeness, which can be finally used to prove the correspondence among $\dot{\approx}_{sb}$, $\dot{\sim}_{sym}^{\Longrightarrow}$ and $\dot{\sim}_I^{\Longrightarrow}$.

3.1.1. A new saturation method

Formally, our new transition relation $\Longrightarrow$ is defined by the rules in Table 6.

Remark 4. *For simplicity, we shall use the same notation ($\Longrightarrow$) we used for Milner's method (Table 5) to denote the new saturation method (Table 6).Consequently the definitions of weak barbs, symbolic and irredundant bisimilarity are now interpreted w.r.t. the new $\Longrightarrow$ as in Table 6 ($\Downarrow$, $\dot{\sim}_{sym}^{\Longrightarrow}$ and $\dot{\sim}_I^{\Longrightarrow}$ respectively).*

R-Tau	$\dfrac{}{\gamma \Longrightarrow \gamma}$	R-Label	$\dfrac{\gamma \xrightarrow{\alpha} \gamma'}{\gamma \overset{\alpha}{\Longrightarrow} \gamma'}$	R-Add	$\dfrac{\gamma \overset{\alpha}{\Longrightarrow} \gamma' \overset{\beta}{\Longrightarrow} \gamma''}{\gamma \overset{\alpha \sqcup \beta}{\Longrightarrow} \gamma''}$

Table 6: New saturation method.

First we show that $\not\downarrow$ coincides with $\Downarrow$ since a transition in $\Longrightarrow$ corresponds to a sequence of reductions.[5]

Lemma 2. $\gamma \longrightarrow^* \gamma'$ *iff* $\gamma \Longrightarrow \gamma'$.

Proof. ($\Rightarrow$) We can decompose $\gamma \longrightarrow^* \gamma'$ as follows $\gamma \longrightarrow \gamma_1 \longrightarrow \ldots \longrightarrow \gamma_i \longrightarrow \gamma'$, now we proceed by induction on i. The base case is $i = 0$, then $\gamma \longrightarrow \gamma'$ and by rule R-Label we have $\gamma \Longrightarrow \gamma'$. For the inductive step, first we have by induction hypothesis that $\gamma \longrightarrow^i \gamma_i$ implies $\gamma \Longrightarrow \gamma_i$ (1), on the other hand, by rule R-Label on $\gamma_i \longrightarrow \gamma'$ we can deduce $\gamma_i \Longrightarrow \gamma'$ (2). Finally by R-Add on (1) and (2) $\gamma \Longrightarrow \gamma'$.

($\Leftarrow$) We proceed by induction on the depth of the inference of $\gamma \Longrightarrow \gamma'$. First, using R-Tau, we can directly conclude $\gamma \longrightarrow^* \gamma$. Secondly, using R-Label, $\gamma \Longrightarrow \gamma'$ implies that $\gamma \longrightarrow \gamma'$. Finally, using R-Add and since $\alpha \sqcup \beta = \mathit{true}$ implies $\alpha = \beta = \mathit{true}$, we get $\gamma \Longrightarrow \gamma'' \Longrightarrow \gamma'$ and by induction hypothesis this means that $\gamma \longrightarrow^* \gamma'' \longrightarrow^* \gamma'$ therefore $\gamma \longrightarrow^* \gamma'$.

□

Using this lemma, it is straightforward to see that the notions of weak barbs coincide.[6]

Lemma 3. $\gamma \Downarrow_e$ *iff* $\gamma \not\downarrow_e$.

Proof. First, let us assume that $\gamma \Downarrow_e$ then by definition $\gamma \longrightarrow^* \gamma' \downarrow_e$, and from Lemma 2 we know that $\gamma \Longrightarrow \gamma' \downarrow_e$, hence $\gamma \not\downarrow_e$. On the other hand, if $\gamma \not\downarrow_e$ then by definition $\gamma \Longrightarrow^* \gamma' \downarrow_e$, if we decompose these transitions then $\gamma \Longrightarrow \ldots \Longrightarrow \gamma'$, and from Lemma 2 $\gamma \longrightarrow^* \ldots \longrightarrow^* \gamma'$, therefore $\gamma \longrightarrow^* \gamma' \downarrow_e$, finally $\gamma \Downarrow_e$. □

As we shall see later on, the lemma above will be used to prove the correspondence between $\dot{\approx}_{sb}$ and $\dot{\sim}_I^{\Longrightarrow}$.

[5]Notice that Lemma 2 also holds for the Milner's saturation method (Table 5)

[6]Notice that Lemma 3 also holds for the Milner's saturation method (Table 5) because of Lemma 2

3.1.2. The new saturation method is finitely branching

An important property that the labeled transition system defined by the new relation $\Longrightarrow$ must fulfill is that it must be finitely branching, given that $\longrightarrow$ is also finitely branching. We prove this next but first let us introduce some useful notation.

The set $\mathsf{Reach}(\gamma, \rightsquigarrow)$, defined below, represents the set of configurations which results after performing one step starting from a given configuration γ and using a relation $\rightsquigarrow$. Such set contains pairs of the form $[\gamma', \alpha]$ in which the first item (γ') is the configuration reached and the second one (α) is the label used for that purpose. Formally we have:

Definition 16 (Single-step Reachable Pairs)**.** *The set of Single-step reachable pairs is defined as* $\mathsf{Reach}(\gamma, \rightsquigarrow) = \{[\gamma', \alpha] \mid \gamma \overset{\alpha}{\rightsquigarrow} \gamma'\}$.

We can extend this definition to consider more than one step at a time. We will call this new set $\mathsf{Reach}^*(\gamma, \rightsquigarrow)$ and it is defined below.

Definition 17 (Reachable Pairs)**.** *The set of reachable pairs is defined as follows:* $\mathsf{Reach}^*(\gamma, \rightsquigarrow) = \{[\gamma', \alpha] \mid \exists \alpha_1, \ldots, \alpha_n.\ \gamma \overset{\alpha_1}{\rightsquigarrow} \ldots \overset{\alpha_n}{\rightsquigarrow} \gamma' \wedge \alpha = \alpha_n\}$.

Using the notation defined above, we shall define formally what we mean by finitely branching as follows.

Definition 18. *We say that a transition relation $\rightsquigarrow$ is finitely branching if for all γ we have $|\mathsf{Reach}(\gamma, \rightsquigarrow)| < \infty$.*

Remark 5. *It is worth noticing that even though we have restricted the syntax of ccp to finite processes, for some constraint systems $\longrightarrow$ may be infinitely-branching due to Rule LR2 in Table 3; i.e., there may be infinitely many minimal labels allowing the transition. For this reason we shall sometimes explicitly assume that $\longrightarrow$ is defined in constraint systems that do not cause $\longrightarrow$ to be infinitely-branching.*

For convenience, in order to project the first or the second item of the reachable pairs we will define the functions $\mathcal{C}$ and $\mathcal{L}$ which, respectively, extract the configuration and the label respectively (hence the name).

Definition 19. *The functions $\mathcal{C}$ and $\mathcal{L}$ are defined as follows, $\mathcal{C}([\gamma, \alpha]) = \{\gamma\}$ and $\mathcal{L}([\gamma, \alpha]) = \{\alpha\}$. They extend to set of pairs as expected, namely given a set of pairs $S = \{p_1, \ldots\}$ then $\mathcal{L}(S) = \bigcup_{p_i \in S} \mathcal{L}(p_i)$ and similarly for $\mathcal{C}$.*

Now, under the assumption that $\longrightarrow$ is finitely branching, we can observe that if the number of configurations that can be reached (in one or more steps) is finite, then the number of labels should also be finite.

Proposition 2. *Suppose $\longrightarrow$ is finitely branching. If $|\mathcal{C}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$ for any γ, then $|\mathcal{L}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$.*

Proof. From the hypothesis we know that there are finite γ' that can be reached, and using the assumption of being finitely branching we can see that from each of those γ' there are only finitely possible α, hence the conclusion. □

Finally, the nature of the labels in CCP is one of the reasons why our new transition system works. The following lemma illustrates the fact that when generating new labels, with the rule R-Add (Table 6) we will not add an infinite number of those. In the following, $C^{*\sqcup}$ will represent the Kleene closure over $\sqcup$ of the set of constraints C.

Lemma 4. *Given a set of constraints C, if $|C| < \infty$ then $|C^{*\sqcup}| < \infty$.*

Proof. Follows from the commutativity and idempotency of $\sqcup$ ($c \sqcup c = c$). □

Using these elements, the finitely branching property of $\Longrightarrow$ follows directly under the assumption that $\longrightarrow$ is finitely branching and our restriction to finite processes (see Remark 5).

Lemma 5. *If $\longrightarrow$ is finitely branching then $\Longrightarrow$ is finitely branching.*

Proof. (1) $|\mathcal{C}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$ for every γ due to our restriction to finite processes and our assumption that $\longrightarrow$ is finitely branching. One can verify that $\mathcal{C}(\mathsf{Reach}^*(\gamma, \Longrightarrow)) = \mathcal{C}(\mathsf{Reach}^*(\gamma, \longrightarrow))$ hence $|\mathcal{C}(\mathsf{Reach}^*(\gamma, \Longrightarrow))| < \infty$.

(2) From the hypothesis and from Proposition 2 $|\mathcal{L}(\mathsf{Reach}^*(\gamma, \longrightarrow))| < \infty$.

(3) One can check that $\mathcal{L}(\mathsf{Reach}(\gamma, \Longrightarrow)) \subseteq \mathcal{L}(\mathsf{Reach}^*(\gamma, \longrightarrow))^{*\sqcup}$ therefore from Lemma 4 and (2) $|\mathcal{L}(\mathsf{Reach}(\gamma, \Longrightarrow))| < \infty$.

From (1) and (3) we can conclude that for any γ, $\Longrightarrow$ is finitely branching. □

3.1.3. A Remark about our Saturation in CCS

We assume that the reader is familiar with CCS [13]. *In this section the transitions, processes and relations are understood in the context of CCS*. It is worth noticing that we could use the saturation method mentioned in the previous section for other formalisms like CCS, but unlike in CCP it would not work as intended: now, the actions that a process can perform need to be sequences and the rules in Table 6 must be replaced by those in Table 7 (essentially, the lub operation $\sqcup$ of CCP is replaced by the concatenation of sequence).

$$\textsf{RCCS1}\quad \frac{}{P \stackrel{\tau}{\Longrightarrow}_{CCS} P} \qquad \textsf{RCCS2}\quad \frac{P \stackrel{s}{\longrightarrow}_{CCS} P'}{P \stackrel{s}{\Longrightarrow}_{CCS} P'}$$

$$\textsf{RCCS3}\quad \frac{P \stackrel{s}{\Longrightarrow}_{CCS} P' \stackrel{s'}{\Longrightarrow}_{CCS} P''}{P \stackrel{s.s'}{\Longrightarrow}_{CCS} P''}$$

Table 7: New Labeled Transition System for CCS. Let $s = a_1 \dots a_n$ be a sequence of observable actions. For the Rule RCCS3 we assume that $s.\tau = \tau.s = s$.

Using these rules we can now define weak bisimilarity in terms of the new relation $\Longrightarrow_{CCS}$ as follows.

Definition 20 (CCS-Weak Bisimilarity)**.** *A symmetric relation $\mathcal{R}$ is a CCS-weak bisimulation if for every $(P, Q) \in \mathcal{R}$:*

- *If $P \stackrel{s}{\Longrightarrow}_{CCS} P'$ then there exists Q' s.t. $Q \stackrel{s}{\Longrightarrow}_{CCS} Q'$ and $(P', Q') \in \mathcal{R}$.*

We say that P and Q are CCS-weakly bisimilar ($P \approx Q$) iff there is a CSS-weak bisimulation containing (P, Q).

This definition resembles the standard definition of strong bisimilarity, hence we could use the procedure to verify the strong version under $\Longrightarrow$ in order to verify the weak one. However, by applying the rules in Table 7, even for a finite LTS, we could end up adding an infinite number of transitions. The following example illustrates the problem.

(a) Original LTS (finitely branching). (b) Saturated LTS (infinitely branching).

Figure 6: CCS Process $P = a.P$ of Example 12

Example 12. *Let us take a typical one-state finitely-branching CCS process with a single transition labeled with a into itself (See Figure 6-(a)). However, if we apply the rules in Table 7 for the example above, we end up adding an infinite number of transitions. Hence, becoming infinitely branching (see Figure 6-(b)).*

3.1.4. Soundness and Completeness

As mentioned before, soundness and completeness of the relation are the core properties when proving $\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_{I}$. We now proceed to show that our method enjoys of these properties and they will allow us to prove the correspondence among the equivalences for the weak case.

Recall that in Definition 12 we introduced the formal definition of completeness, now we shall introduce the notion of soundness.

Definition 21 (Soundness)**.** *We say that $\rightsquigarrow$ is sound iff whenever $\langle P, c\rangle \overset{\alpha}{\rightsquigarrow} \langle P', c'\rangle$ then $\langle P, c \sqcup \alpha\rangle \rightsquigarrow \langle P', c'\rangle$.*

Below we shall prove that $\Longrightarrow$ from Table 6 is sound and complete.

Lemma 6 (Soundness of $\Longrightarrow$)**.** *If $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$ then $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$.*

Proof. We proceed by induction on the depth of the inference of $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$.

- Using R-Tau we have $\langle P, c\rangle \Longrightarrow \langle P, c\rangle$ and the result follows directly given that $\alpha = true$.

- Using R-Label we have $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c'\rangle$ then $\langle P, c\rangle \stackrel{\alpha}{\longrightarrow} \langle P', c'\rangle$. By Lemma 1 (soundness of $\longrightarrow$) we get $\langle P, c \sqcup \alpha\rangle \longrightarrow \langle P', c'\rangle$ and finally by rule R-Label $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$.

- Using R-Add then we have $\langle P, c\rangle \stackrel{\beta\sqcup\lambda}{\Longrightarrow} \langle P', c'\rangle$ then $\langle P, c\rangle \stackrel{\beta}{\Longrightarrow} \langle P'', c''\rangle \stackrel{\lambda}{\Longrightarrow} \langle P', c'\rangle$ where $\beta \sqcup \lambda = \alpha$. By induction hypothesis, $\langle P, c \sqcup \beta\rangle \Longrightarrow \langle P'', c''\rangle$ (1) and $\langle P'', c'' \sqcup \lambda\rangle \Longrightarrow \langle P', c'\rangle$ (2). By monotonicity on (1), $\langle P, c \sqcup \beta \sqcup \lambda\rangle \Longrightarrow \langle P'', c'' \sqcup \lambda\rangle$ and by rule R-Add on this transition and (2) then, given that $\beta \sqcup \lambda = \alpha$, we obtain $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$.

□

Lemma 7 (Completeness of $\Longrightarrow$)**.** *If $\langle P, c \sqcup a\rangle \Longrightarrow \langle P', c'\rangle$ then there exist α and b s.t. $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$.*

Proof. Assuming that $\langle P, c \sqcup a\rangle \Longrightarrow \langle P', c'\rangle$ then, from Lemma 2, we can say that $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'\rangle$ which can be written as $\langle P, c \sqcup a\rangle \longrightarrow \ldots \longrightarrow \langle P_i, c_i\rangle \longrightarrow \langle P', c'\rangle$, we will proceed by induction on i.

- (Base Case) Assuming $i = 0$ then $\langle P, c \sqcup a\rangle \longrightarrow \langle P', c'\rangle$ and the result follows directly from Lemma 1 (Completeness of $\longrightarrow$) and R-Label.

- (Induction) Let us assume that $\langle P, c \sqcup a\rangle \longrightarrow^i \langle P_i, c_i\rangle \longrightarrow \langle P', c'\rangle$ then by induction hypothesis there exist β and b' s.t. $\langle P, c\rangle \stackrel{\beta}{\Longrightarrow} \langle P_i, c_i'\rangle$ (1) where $\beta \sqcup b' = a$ and $c_i' \sqcup b' = c_i$. Now by completeness on the last transition $\langle P_i, \overbrace{c_i}^{c_i' \sqcup b'}\rangle \longrightarrow \langle P', c'\rangle$, there exists λ and b'' s.t. $\langle P_i, c_i'\rangle \stackrel{\lambda}{\longrightarrow} \langle P', c''\rangle$ where $\lambda \sqcup b'' = b'$ and $c'' \sqcup b'' = c'$, thus by rule R-Label we have $\langle P_i, c_i'\rangle \stackrel{\lambda}{\Longrightarrow} \langle P', c''\rangle$ (2). We can now proceed to apply rule R-Add on (1) and (2) to obtain the transition $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c''\rangle$ where $\alpha = \beta \sqcup \lambda$ and finally take $b = b''$, therefore the conditions hold $\alpha \sqcup b = \beta \sqcup \lambda \sqcup b'' = a$ and $c'' \sqcup b = c'' \sqcup b'' = c'$.

□

3.2. Deciding Observational Equivalence

In this section we shall show our main result, a method for deciding $\dot{\approx}_{sb}$. Recall that $\dot{\approx}_{sb}$ is the standard weak bisimilarity for CCP [10], and it is defined in terms of $\longrightarrow$, therefore it does not depend on $\Longrightarrow$. We start from the fact that

using CCP partition refinement (Section 2.3.2) one can check whether two configurations are irredundant bisimilar ($\dot{\sim}_I$). Following the same approach, we prove that $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$ hence we give a reduction from $\dot{\approx}_{sb}$ to $\dot{\sim}^{\Longrightarrow}_{I}$. Finally, $\dot{\sim}^{\Longrightarrow}_{I}$ can be verified using a modified version of the CCP partition refinement (as we shall see in Section 4).

First, given that $\Longrightarrow$ is sound and complete (Lemma 6 and Lemma 7), the correspondence between the symbolic and irredundant bisimilarity follows from [12].

Corollary 1. *$\gamma \dot{\sim}^{\Longrightarrow}_{sym} \gamma'$ iff $\gamma \dot{\sim}^{\Longrightarrow}_{I} \gamma'$*

Finally, in the next two lemmata, we prove that $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym}$.

Lemma 8. *If $\gamma \dot{\approx}_{sb} \gamma'$ then $\gamma \dot{\sim}^{\Longrightarrow}_{sym} \gamma'$*

Proof. We need to prove that $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle\}$ is a symbolic bisimulation over $\Longrightarrow$. The first condition (i) of the bisimulation follows directly from Lemma 3. As for (ii), let us assume that $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$ then by soundness of $\Longrightarrow$ we have $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', c'\rangle$, now by Lemma 2 we obtain $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', c'\rangle$. Given that $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$ then from the latter transition we can conclude that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle P', c'\rangle \dot{\approx}_{sb} \langle Q', d'\rangle$, hence we can use Lemma 2 again to deduce that $\langle Q, d \sqcup \alpha\rangle \Longrightarrow \langle Q', d'\rangle$. Finally, by completeness of $\Longrightarrow$, there exist β and b s.t. $t = \langle Q, d\rangle \overset{\beta}{\Longrightarrow} \langle Q', d''\rangle$ where $\beta \sqcup b = \alpha$ and $d'' \sqcup b = d'$, therefore $t \vdash_D \langle Q, d\rangle \overset{\alpha}{\rightsquigarrow} \langle Q', d'\rangle$ and $\langle P', c'\rangle \mathcal{R} \langle Q', d'\rangle$. □

Lemma 9. *If $\gamma \dot{\sim}^{\Longrightarrow}_{sym} \gamma'$ then $\gamma \dot{\approx}_{sb} \gamma'$*

Proof. We need to prove that $\mathcal{R} = \{(\langle P, c \sqcup a\rangle, \langle Q, d \sqcup a\rangle) \mid \langle P, c\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q, d\rangle\}$ is a weak saturated bisimulation. First, condition (i) follows form Lemma 3 and (iii) by definition of $\mathcal{R}$. Let us prove condition (ii), assume $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'\rangle$ then by Lemma 2 $\langle P, c \sqcup a\rangle \Longrightarrow \langle P', c'\rangle$. Now by completeness of $\Longrightarrow$ there exist α and b s.t. $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c''\rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$. Since $\langle P, c\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q, d\rangle$ then we know there exists a transition $t = \langle Q, d\rangle \overset{\beta}{\Longrightarrow} \langle Q', d'\rangle$ s.t. $t \vdash_D \langle Q, d\rangle \overset{\alpha}{\rightsquigarrow} \langle Q', d''\rangle$ and $(\mathbf{a})\langle P', c''\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q', d''\rangle$, by definition of $\vdash_D$ there exists b' s.t. $\beta \sqcup b' = \alpha$ and $d' \sqcup b' = d''$. Using soundness of $\Longrightarrow$ on t we get $\langle Q, d \sqcup \beta\rangle \Longrightarrow \langle Q', d'\rangle$, thus by Lemma 2 $\langle Q, d \sqcup \beta\rangle \longrightarrow^* \langle Q', d'\rangle$ and finally by monotonicity

$$\langle Q, d \sqcup \overbrace{\underbrace{\beta \sqcup b'}_{\alpha} \sqcup b}^{a}\rangle \longrightarrow^* \langle Q', \overbrace{d' \sqcup b'}^{d''} \sqcup b\rangle \tag{2}$$

Then, the transition $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'\rangle$ can be rewritten as $\langle P, c \sqcup a\rangle \longrightarrow^* \langle P', c'' \sqcup b\rangle$, and using (2), $\langle Q, d \sqcup a\rangle \longrightarrow^* \langle Q', d'' \sqcup b\rangle$. It is left to prove that $\langle P', c'' \sqcup b\rangle \mathcal{R} \langle Q', d'' \sqcup b\rangle$ which follows from $(\mathbf{a})$ and Theorem 1. □

Using Lemma 8 and Lemma 9 we obtain the following theorem.

Theorem 3. $\langle P, c\rangle \dot{\sim}^{\Longrightarrow}_{sym} \langle Q, d\rangle$ *iff* $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$

From the above results, we conclude that $\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{I}$.

4. Algorithm for observational equivalence in CCP

In this section we describe the decision procedure for verifying weak version of saturated bisimilarity ($\dot{\approx}_{sb}$). The approach is to reduce the problem of deciding $\dot{\approx}_{sb}$ to that of deciding strong bisimilarity as illustrated in Figure 7. The latter problem can be decided using the partition refinement for CCP (Algorithm 2).

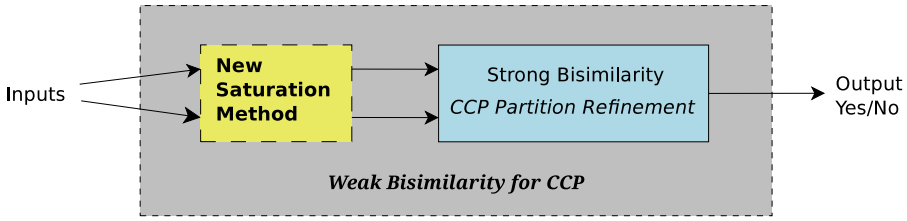

Figure 7: Summary of our approach

More precisely, given two configurations γ and γ', the first step is to build $G = LTS_{\longrightarrow}(IS)$ where $IS = \{\gamma, \gamma'\}$. We proceed to compute $G' = LTS_{\Longrightarrow}(IS)$, and then run an adaptation of Algorithm 2 on G'. The adaptation consists in using weak barbs ($\Downarrow_c$) instead of barbs ($\downarrow_c$) for the initial partition $\mathcal{P}^0$, and in using $\Longrightarrow$ as a parameter of Algorithm 3.

Using this algorithm we can decide $\dot{\approx}_{sb}$ also with exponential time as we shall see below. First, let us determine the complexity of computing $LTS_{\Longrightarrow}(IS)$.

Proposition 3. *Assume that* $\longrightarrow$ *is finitely branching (Definition 18). Let* $G = LTS_{\longrightarrow}(IS)$ *and* $G' = LTS_{\Longrightarrow}(IS)$. *Let* $N = |\mathtt{V}(G)| = |\mathtt{E}(G)|$, $N' = |\mathtt{V}(G')|$ *and* $M' = |\mathtt{E}(G)|$. *We have that* $N' = N$ *and* $M' = O(N^2)$.

Proof. By construction of γ' the rules in Table 6 never add a new vertex, thus it follows directly that $N' = N$. As for the edges, the rules in Table 6 will add, at most, a transition from each element in $\mathtt{V}(G)$ to every other configuration in $\mathtt{V}(G)$. Since $\mathtt{V}(G) = N$ then the resulting transitions are $M' = O(N^2)$. □

Algorithm 3 `weak-pr-ccp`$(IS, \rightsquigarrow)$

Initialization

1. Compute $IS^{\star}_{\rightsquigarrow}$ with the rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ defined in Table 4,
2. $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$ is a partition of $IS^{\star}_{\rightsquigarrow}$ where γ and γ' are in B_i iff they satisfy the same *weak barbs* ($\Downarrow_c$),

Iteration $\mathcal{P}^{n+1} := \mathbf{IR}_{\rightsquigarrow}(\mathcal{P}^n)$ as in Definition 11
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

We now show that the size of $IS^{\star}_{\Longrightarrow}$ may be exponential w.r.t. $|\mathtt{Config}_{\longrightarrow}(IS)|$.

Lemma 10. *There exists IS such that* $|IS^{\star}_{\Longrightarrow}| = \Omega(2^{|\mathtt{Config}_{\longrightarrow}(IS)|})$.

Proof. Let $n > 0$. We define $P^n = P^n_0$ with P^n_i, for $i \in \{0, \ldots, n-1\}$, given by:

$$P^n_i = (\mathbf{ask}\ b_i\ \rightarrow\ \mathbf{stop}) + (\mathbf{ask}\ a_i\ \rightarrow\ P^n_{i+1})$$

and $P^n_n = \mathbf{tell}(b_n)$. Furthermore, we assume that for all $i \in \{0, \ldots, n-1\}$ we have $a_i \sqsubset b_i$ and for all $j \in \{0, \ldots, n-1\}$ if $i \neq j$ then $a_i \not\sqsubseteq a_j$ and $b_i \not\sqsubseteq b_j$. The reachable states from $\langle P^n, \mathit{true}\rangle$ are illustrated below.

$$\begin{array}{ccc}
\langle P^n, \mathit{true}\rangle & \xrightarrow{\quad b_0 \quad} & \langle \mathbf{stop}, b_0\rangle \\
\downarrow a_0 & & \\
\langle P^n_1, a_0\rangle & \xrightarrow{\quad b_1 \quad} & \langle \mathbf{stop}, a_0 \sqcup b_1\rangle \\
\downarrow a_1 & & \\
\cdots & & \\
\downarrow a_{n-1} & & \\
\langle P^n_n, \bigsqcup_{i=0}^{n-1} a_i\rangle & \xrightarrow{\quad b_n \quad} & \langle \mathbf{stop}, b_n \sqcup \bigsqcup_{i=0}^{n-1} a_i\rangle
\end{array}$$

Notice that that size of the set of reachable states $|\mathtt{Config}_{\longrightarrow}(IS)|$ is $2n$. By using the rules $(\mathsf{IS}^{IS}_{\rightsquigarrow})$, $(\mathsf{RS}^{IS}_{\rightsquigarrow})$ and $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ in Table 4 with $\rightsquigarrow = \Longrightarrow$ and also $IS = \{\langle P^n, \mathit{true}\rangle\}$, we obtain an $IS^{\star}_{\Longrightarrow}$ whose size is given by the following recurrence relation, for $n > 0$: $f(n) = 2f(n-1) + 2$ with $f(0) = 2$. Without loss of generality consider the first level of transitions of $\langle P^n, \mathit{true}\rangle$:

$$\begin{array}{ccc} \langle P^n, true\rangle & \xrightarrow{\quad b_0 \quad} & \langle \mathbf{stop}, b_0\rangle \\ \Big\downarrow a_0 & & \\ \langle P_1^n, a_0\rangle & & \end{array}$$

First, we count $\langle P^n, true\rangle$ and $\langle \mathbf{stop}, b_0\rangle$, hence the $+2$ in $f(n)$. Furthermore, the rest of the states are those generated by $\langle P_1^n, a_0\rangle$ and these are exactly $f(n-1)$. Moreover, since $a_0 \sqsubset b_0$ then by rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ a new state $\langle P_1^n, b_0\rangle$ is added to $IS^{\star}_{\Longrightarrow}$. Note that $\langle P_1^n, b_0\rangle$ produces the same number of states as $\langle P_1^n, a_0\rangle$ since by construction a_0 and b_0 are irrelevant in P_1^n. Therefore we count the states generated by $\langle P_1^n, b_0\rangle$ as another $f(n-1)$. Also note that the same reasoning applies at every level of the process. By solving the recurrence we can conclude that $f(n) = \Omega(2^n)$ hence $|IS^{\star}_{\Longrightarrow}| = \Omega(2^n)$ as wanted. □

We can now state correctness and complexity of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$.

Theorem 4. *Assume that $\longrightarrow$ is finitely branching (Definition 18). Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$ in Algorithm 3. Then*

- *$\gamma\ \mathcal{P}\ \gamma'$ iff $\gamma \mathrel{\dot{\approx}_{sb}} \gamma'$.*
- *$\texttt{weak-pr-ccp}(IS, \Longrightarrow)$ may take exponential time in $|\texttt{Config}_{\longrightarrow}(IS)|$.*

Proof. The first point follows from Corollary 1 ($\dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$) and Theorem 3 ($\dot{\sim}^{\Longrightarrow}_{sym} = \dot{\approx}_{sb}$). For the second point, as for the strong case, the exponential time is due to construction of the set $IS^{\star}_{\Longrightarrow}$ in step 1 of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$, whose size is exponential in $|\texttt{Config}_{\longrightarrow}(IS)|$ as shown in Lemma 10). □

Implementation. We conclude by showing some experiments performed using our tool. In Figure 8 we present the results obtained after running two sets of experiments.

In Figure 8-(a) we evaluate the performance of our procedure in the worst-case scenario described in Lemma 10, which is exponential in $|\texttt{Config}_{\longrightarrow}(IS)|$. Moreover, because of Proposition 3, we know that the size of $LTS_{\Longrightarrow}(IS)$ can grow (at most) quadratically w.r.t. $|\texttt{Config}_{\longrightarrow}(IS)|$. Consequently, we can see that for $\texttt{Nodes} = 10$ the algorithm for weak bisimilarity already takes more than the strong one with $\texttt{Nodes} = 40$.

In Figure 8-(b) we focused on a set of cases in which $LTS_{\Longrightarrow}(IS)$ grows linearly w.r.t. $|\texttt{Config}_{\longrightarrow}(IS)|$. The idea is to evaluate the performance of the weak

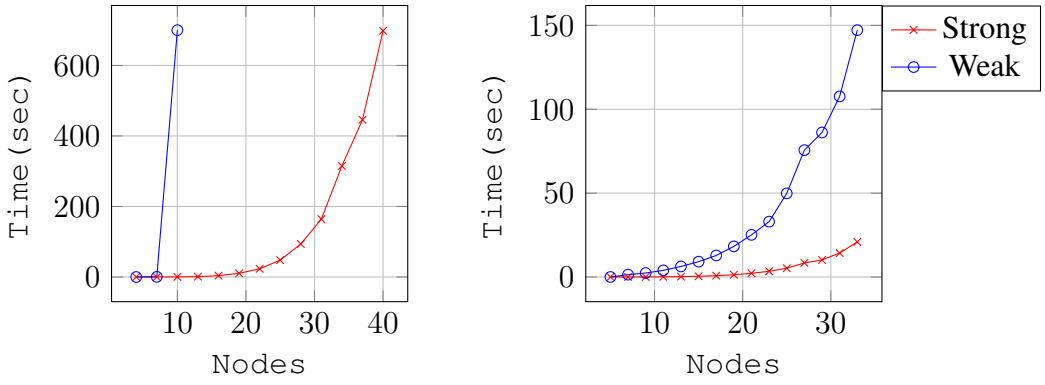

Figure 8: Nodes is the size of $\texttt{Config}_{\longrightarrow}(IS)$ for a given set of initial configurations IS. In (a) we tested the case in which $LTS_{\Longrightarrow}(IS)$ grows quadratically w.r.t. Nodes by constructing IS as in the worst-case scenario described in Lemma 10. In (b) we generated IS where the size of $LTS_{\Longrightarrow}(IS)$ grows linearly w.r.t. Nodes.

procedure in a better scenario in order to compare it with the strong version. We did this by randomly generating test cases fulfilling the linearity constraint.

5. Concluding Remarks

We showed that the weak transition relation using Milner's saturation method is not complete for CCP (in the sense of Definition 12). This implied that $\dot{\approx}_{sb}$ (Definition 4, [10]) cannot be computed immediately by using the CCP partition refinement algorithm for (strong) bisimilarity CCP on the saturated transition relation. We then introduced a new transition relation using a different saturation mechanism and showed that it is complete for CCP and also that it is finitely branching. As a consequence, we also showed that the CCP partition refinement can be used to compute $\dot{\approx}_{sb}$ using this new relation. Likewise, we have shown that although this new saturation method elaborated for CCP could be used for any other formalisms such as CCS, it would not work as desired because it could transform a finitely branching LTS into an infinitely branching one. To the best of our knowledge, this is the first approach to verifying weak bisimilarity for CCP. As future work, we plan to adapt and implement our notion of (weak) bisimilarity for variants of CCP, in particular for Spatial CCP [31]. This calculus is a meaningful extension of CCP for modeling spatially distributed concurrent systems such as social networks. The spatial and hierarchical aspects of this variant may offer

challenges for the theoretical and practical aspects of a weak bisimulation semantics.

Acknowledgments. We would like to thank the anonymous reviewers of this paper. Their constructive criticism and suggestions helped us improve the previous versions of this paper.

References

[1] I. Lanese, J. A. Pérez, D. Sangiorgi, A. Schmitt, On the expressiveness and decidability of higher-order process calculi, Information and Computation 209 (2011) 198–226. doi:10.1016/j.ic.2010.10.001.

[2] B. Victor, F. Moller, The mobility workbench - a tool for the π-calculus, in: D. L. Dill (Ed.), 6th International Conference on Computer Aided Verification (CAV 1994), volume 818 of *Lecture Notes in Computer Science*, Springer, 1994, pp. 428–440. doi:10.1007/3-540-58179-0_73.

[3] J.-C. Fernandez, An implementation of an efficient algorithm for bisimulation equivalence, Science of Computer Programming 13 (1989) 219–236. doi:10.1016/0167-6423(90)90071-K.

[4] G. Ferrari, S. Gnesi, U. Montanari, M. Pistore, G. Ristori, Verifying mobile processes in the HAL environment, in: A. J. Hu, M. Y. Vardi (Eds.), 10th International Conference on Computer Aided Verification (CAV 1998), volume 1427 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 511–515. doi:10.1007/BFb0028772.

[5] P. C. Kanellakis, S. A. Smolka, CCS expressions, finite state processes, and three problems of equivalence, in: R. L. Probert, N. A. Lynch, N. Santoro (Eds.), 2nd Annual ACM Symposium on Principles of Distributed Computing (PODC 1983), ACM, 1983, pp. 228–240. doi:10.1145/800221.806724.

[6] R. Paige, R. E. Tarjan, Three partition refinement algorithms, SIAM Journal on Computing 16 (1987) 973–989. doi:10.1137/0216062.

[7] V. A. Saraswat, M. C. Rinard, Concurrent constraint programming, in: F. E. Allen (Ed.), 17th Annual ACM Symposium on Principles of Programming Languages (POPL 1990), ACM Press, 1990, pp. 232–245. doi:10.1145/96709.96733.

[8] F. Bonchi, B. König, U. Montanari, Saturated semantics for reactive systems, in: 21th IEEE Symposium on Logic in Computer Science (LICS 2006), IEEE Computer Society, 2006, pp. 69–80. doi:10.1109/LICS.2006.46.

[9] F. Bonchi, F. Gadducci, G. V. Monreale, Reactive systems, barbed semantics, and the mobile ambients, in: L. de Alfaro (Ed.), 12th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2009), volume 5504 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 272–287. doi:10.1007/978-3-642-00596-1_20.

[10] A. Aristizábal, F. Bonchi, C. Palamidessi, L. Pino, F. D. Valencia, Deriving labels and bisimilarity for concurrent constraint programming, in: M. Hofmann (Ed.), 14th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2011), volume 6604 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 138–152. doi:10.1007/978-3-642-19805-2_10.

[11] V. A. Saraswat, M. C. Rinard, P. Panangaden, Semantic foundations of concurrent constraint programming, in: D. S. Wise (Ed.), 18th Annual ACM Symposium on Principles of Programming Languages (POPL 1991), ACM Press, 1991, pp. 333–352. doi:10.1145/99583.99627.

[12] A. Aristizábal, F. Bonchi, L. Pino, F. D. Valencia, Partition refinement for bisimilarity in CCP, in: S. Ossowski, P. Lecca (Eds.), 27th Annual ACM Symposium on Applied Computing (SAC 2012), ACM, 2012, pp. 88–93. doi:10.1145/2245276.2245296.

[13] R. Milner, A Calculus of Communicating Systems, volume 92 of *Lecture Notes in Computer Science*, Springer, 1980. doi:10.1007/3-540-10235-3.

[14] A. Aristizábal, F. Bonchi, L. F. Pino, F. Valencia, Reducing weak to strong bisimilarity in CCP, in: M. Carbone, I. Lanese, A. Silva, A. Sokolova (Eds.), 5th Interaction and Concurrency Experience (ICE 2012), volume 104 of *Electronic Proceedings in Theoretical Computer Science*, 2012, pp. 2–16. doi:10.4204/EPTCS.104.2.

[15] P. Baldan, A. Bracciali, R. Bruni, A semantic framework for open processes, Theoretical Computer Science 389 (2007) 446–483. doi:10.1016/j.tcs.2007.09.004.

[16] P. Sobocinski, Representations of petri net interactions, in: P. Gastin, F. Laroussinie (Eds.), 21th International Conference on Concurrency Theory (CONCUR 2010), volume 6269 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 554–568. doi:10.1007/978-3-642-15375-4_38.

[17] R. Bruni, H. C. Melgratti, U. Montanari, A connector algebra for p/t nets interactions, in: J.-P. Katoen, B. König (Eds.), 22nd International Conference on Concurrency Theory (CONCUR 2011), volume 6901 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 312–326. doi:10.1007/978-3-642-23217-6_21.

[18] F. Gadducci, U. Montanari, The tile model, in: G. D. Plotkin, C. Stirling, M. Tofte (Eds.), Proof, Language, and Interaction, Essays in Honour of Robin Milner, The MIT Press, 2000, pp. 133–166.

[19] R. Bruni, F. Gadducci, U. Montanari, P. Sobocinski, Deriving weak bisimulation congruences from reduction systems, in: M. Abadi, L. de Alfaro (Eds.), 16th International Conference on Concurrency Theory (CONCUR 2005), volume 3653 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 293–307. doi:10.1007/11539452_24.

[20] P. Sewell, From rewrite rules to bisimulation congruences, Theoretical Computer Science 274 (2002) 183–230. doi:10.1016/S0304-3975(00)00309-1.

[21] J. J. Leifer, R. Milner, Transition systems, link graphs and petri nets, Mathematical Structures in Computer Science 16 (2006) 989–1047. doi:10.1017/S0960129506005664.

[22] O. H. Jensen., Mobile Processes in Bigraphs, Ph.D. thesis, University of Cambridge, 2006.

[23] P. Sobocinski, Relational presheaves as labelled transition systems, in: D. Pattinson, L. Schröder (Eds.), 11th International Workshop on Coalgebraic Methods in Computer Science (CMCS 2012), volume 7399 of *Lec-*

ture Notes in Computer Science, Springer, 2012, pp. 40–50. doi:10.1007/978-3-642-32784-1_3.

[24] J. J. Leifer, R. Milner, Deriving bisimulation congruences for reactive systems, in: C. Palamidessi (Ed.), 11th International Conference on Concurrency Theory (CONCUR 2005), volume 1877 of *Lecture Notes in Computer Science*, Springer, 2000, pp. 243–258. doi:10.1007/3-540-44618-4_19.

[25] D. Sangiorgi, J. Rutten, Advanced Topics in Bisimulation and Coinduction, Cambridge University Press, 2012. doi:10.1017/CBO9780511792588.

[26] F. S. de Boer, A. D. Pierro, C. Palamidessi, Nondeterminism and infinite computations in constraint programming, Theoretical Computer Science 151 (1995) 37–78. doi:10.1016/0304-3975(95)00047-Z.

[27] N. P. Mendler, P. Panangaden, P. J. Scott, R. A. G. Seely, A logical view of concurrent constraint programming, Nordic Journal of Computing 2 (1995) 181–220.

[28] J. Monk, L. Henkin, A. Tarski., Cylindric Algebras (Part I), North-Holland, 1971.

[29] R. Milner, D. Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), 19th International Colloquium on Automata, Languages and Programming (ICALP 1992), volume 623 of *Lecture Notes in Computer Science*, Springer, 1992, pp. 685–695. doi:10.1007/3-540-55719-9_114.

[30] F. Bonchi, U. Montanari, Minimization algorithm for symbolic bisimilarity, in: G. Castagna (Ed.), 18th European Symposium on Programming (ESOP 2009), volume 5502 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 267–284. doi:10.1007/978-3-642-00590-9_20.

[31] S. Knight, C. Palamidessi, P. Panangaden, F. D. Valencia, Spatial and epistemic modalities in constraint-based process calculi, in: M. Koutny, I. Ulidowski (Eds.), 23rd International Conference on Concurrency Theory (CONCUR 2012), volume 7454 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 317–332. doi:10.1007/978-3-642-32940-1.