Efficient Algorithms for Program Equivalence for Confluent Concurrent Constraint Programming

Luis Fernando Pino Duque, Filippo Bonchi, Frank Valencia

Efficient Algorithms for Program Equivalence for Confluent Concurrent Constraint Programming[☆]

Luis F. Pino

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

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 *declarative* framework from concurrency theory. Its foundations and principles e.g., semantics, proof systems, axiomatizations, have been thoroughly studied for over the last two decades. In contrast, the development of algorithms and automatic verification procedures for CCP have hitherto been far too little considered. To the best of our knowledge there is only one existing verification algorithm for the standard notion of CCP program (observational) equivalence. In this paper we first show that this verification algorithm has an *exponential-time* complexity even for programs from a representative sub-language of CCP; the *summation-free fragment* (CCP$\backslash$+). We then significantly improve on the complexity of this algorithm by providing two alternative *polynomial-time* decision procedures for CCP$\backslash$+ program equivalence. Each of these two procedures has an advantage over the other. One has a better time complexity. The other can be easily adapted for the full

[☆]This work has been partially supported by the project ANR 12IS02001 PACE, 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), filippo.bonchi@ens-lyon.fr (Filippo Bonchi), frank.valencia@lix.polytechnique.fr (Frank Valencia)

language of CCP to produce significant state space reductions. The relevance of both procedures derives from the importance of CCP\+. This fragment, which has been the subject of many theoretical studies, has strong ties to first-order logic and an elegant denotational semantics, and it can be used to model real-world situations. Its most distinctive feature is that of *confluence*, a property we exploit to obtain our polynomial procedures. Finally, we also study the congruence issues regarding CCP's program equivalence.

Keywords: Concurrent Constraint Programming, Bisimulation, Partition Refinement, Observational Equivalence

---

1. Introduction

Motivation. *Concurrent constraint programming (CCP)* [1] is a well-established *formalism* from concurrency theory that combines the traditional algebraic and operational view of process calculi with a *declarative* one based upon logic. It was designed to give programmers explicit access to the concept of partial information and, as such, has close ties with *logic and constraint programming*.

The CCP framework models systems whose agents (processes or programs) interact by concurrently *posting* (telling) and *querying* (asking for) partial information in a shared medium (the store). This framework is parametric in a *constraint system* indicating interdependencies (entailment) between partial information and providing for the specification of data types and other rich structures. The above features have attracted renewed attention as witnessed by the works [2, 3, 4, 5, 6] on calculi exhibiting data-types, logic assertions as well as tell and ask operations. A compelling example of the kind of system CCP can model involves users interacting by posting and querying information in a social network [6].

Nevertheless, despite the extensive research on the foundations and principles of CCP, the development of tools and algorithms for the automatic verification of CCP programs has hitherto been far too little considered. As we shall argue below, the only existing algorithm for deciding the standard notion of process equivalence was given in [7] and it has an *exponential time* (and space) complexity.

The main goal of this paper is to produce efficient decision procedures for program equivalence for a meaningful fragment of CCP. Namely, the *summation-free fragment of CCP*, henceforth CCP\+. The CCP\+ formalism is perhaps the most representative sublanguage of CCP. It has been the subject of many theoretical studies because of its computational expressivity, strong ties to first-order logic,

and elegant denotational semantics based on closure operators [1]. Its most distinctive property is that of *confluence* in the sense that the final resulting store is the same regardless of the execution order of the parallel processes. We shall use this property extensively in proving the correctness of our decision procedures.

Approach. To explain our approach we shall briefly recall some CCP equivalences. The standard notion of *observational (program) equivalence*, $\sim_o$, [1], roughly speaking, decrees that two CCP programs are observationally equivalent if each one can be replaced with the other in any CCP context and produce the same final stores. Other alternative notions of program equivalences for CCP such as saturated barbed bisimilarity ($\dot{\sim}_{sb}$) and its weak variant ($\dot{\approx}_{sb}$) were introduced in [8, 9], where it is also shown that $\dot{\approx}_{sb}$ *coincides* with the standard CCP observational equivalence for CCP\+ programs.

The above-mentioned alternative notions of CCP equivalences are defined in terms of a *labeled transition system* (LTS) describing the interactive behavior of CCP programs. (Intuitively, a labeled transition $\gamma \xrightarrow{\alpha} \gamma'$ represents the evolution into the program configuration γ' if the information α is added to store of the program configuration γ.) The advantage of using these alternative notions of equivalence instead of using directly the standard notion of observational equivalence for CCP is that there is a substantial amount of work supporting the automatic verification of bisimilarity-based equivalence.

Unfortunately, the standard algorithms for checking bisimilarity (such as [10, 11, 12, 13]) cannot be reused for $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$, since in this particular case of the bisimulation game, when the attacker proposes a transition, the defender does not necessarily have to answer with a transition with the same label. (This is analogous to what happens in the asynchronous π-calculus [14] where an input transition can be matched also by an internal (tau) transition.)

Partition Refinement for CCP. By building upon [14], we introduced in [15] a variation of the partition refinement algorithm that allows us to decide $\dot{\sim}_{sb}$ in CCP. The variation is based on the observation that some of the transitions are *redundant*, in the sense that they are logical consequences of other transitions. Unfortunately, such a notion of redundancy is not syntactic, but semantic, more precisely, it is based on $\dot{\sim}_{sb}$ itself. Now, if we consider the transition system having only non-redundant transitions, the ordinary notion of bisimilarity coincides with $\dot{\sim}_{sb}$. Thus, in principle, we could remove all the redundant transitions and then check bisimilarity with a standard algorithm. But how can we decide which transitions are redundant, if redundancy itself depends on $\dot{\sim}_{sb}$?

The solution in [15] consists in computing $\dot{\sim}_{sb}$ and redundancy *at the same time*. In the first step, the algorithm considers all the states as equivalent and all the (potentially redundant) transitions as redundant. In any iteration, states are discerned according to (the current estimation of) non-redundant transitions and then non-redundant transitions are updated according to the new computed partition.

One peculiarity of the algorithm in [15] is that in the initial partition, we insert not only the reachable states, but also extra ones which are needed to check for redundancy. Unfortunately, the number of these states might be exponentially bigger than the size of the original LTS and therefore worst-case complexity is *exponential*, even as we shall show in this paper, for the restricted case of CCP\+.

This becomes even more problematic when considering the weak semantics $\dot{\approx}_{sb}$. Usually weak bisimilarity is computed by first closing the transition relation with respect to internal transitions and then by checking strong bisimilarity on the obtained LTS. In [7], this approach (which is referred in [16] as saturation) is proven to be unsound for CCP (see [7]). It is also shown that in order to obtain a sound algorithm, one has to close the transition relation, not only w.r.t. the internal transitions, but w.r.t. *all* the transitions. This induces an explosion of the number of transitions which makes the computation of $\dot{\approx}_{sb}$ even more inefficient.

Confluent CCP. In this paper, we shall consider the "summation free" fragment of CCP (CCP\+), i.e., the fragment of CCP without non-deterministic choice. Differently from similar fragments of other process calculi (such as the π-calculus or the mobile ambients), CCP\+ is *confluent* because concurrent constraints programs interact only via reading and telling permanent pieces of information (roughly speaking, resources are not consumed). When considering the weak equivalence $\dot{\approx}_{sb}$, confluence makes it possible to characterize redundant transitions syntactically, i.e., without any information about $\dot{\approx}_{sb}$. Therefore for checking $\dot{\approx}_{sb}$ in CCP\+, we can first prune redundant transitions and then check the standard bisimilarity with one of the usual algorithms [10, 11, 12, 13]. Since redundancy can be determined statically, the additional states needed by the algorithm in [15] are not necessary any more: in this way, the worst case complexity from exponential becomes *polynomial*.

Unfortunately, this approach still suffers of the explosion of transitions caused by the closure of the transition relation. In order to avoid this problem, we exploit a completely different approach (based on the semantic notion of *compact input-output sets*) that works directly on the original LTS. We shall conclude our paper by also showing how the results obtained for CCP\+can be exploited to optimize

the partition refinement for the full language of CCP.

We wish to conclude this introduction with a quote from [17] that captures the goal of the present paper:

"*The times have gone, where formal methods were primarily a pen-and-pencil activity for mathematicians. Today, only languages properly equipped with software tools will have a chance to be adopted by industry. It is therefore essential for the next generation of languages based on process calculi to be supported by compilers, simulators, verification tools, etc. The research agenda for theoretical concurrency should therefore address the design of efficient algorithms for translating and verifying formal specifications of concurrent systems*".

Contributions. This paper is an extended version of [18]. In this version we give all the details and proofs omitted in [18]. We also extend the intuitions from [18] to improve the clarity of the paper. Furthermore, we add a section, Section 7, with new technical material. In this new section we study congruence issues for the CCP fragment here studied, CCP\+, as well as for the full language of CCP. In particular we show that $\dot{\approx}_{sb}$ is a congruence for CCP\+ but not for CCP. Despite the negative result for the full language we show that $\dot{\approx}_{sb}$ is still a useful notion as it implies the standard notion of program equivalence $\sim_o$ for CCP. Consequently, the (co-inductive) proof techniques associated to $\dot{\approx}_{sb}$ can be used to establish program equivalence and the algorithms provided in this paper for the full language can be used as semi-decision procedures for $\sim_o$.

Structure of the paper. The paper is organized as follows: In Section 2 we recall the background concerning the standard partition refinement and the CCP formalism. In Section 3 we present the partition refinement for CCP from [15] and how it can be used to decide observational equivalence following [7]. Our contributions begin in Section 4 where we prove that the partition refinement for CCP from Section 3 is inefficient even for CCP\+. We then introduce some particular features of CCP\+ which are then used to develop a polynomial procedure for checking observational equivalence in CCP\+. In Section 5 we introduce our second, more efficient, method for deciding observational equivalence by using the compact input-output sets. In Section 6 we show how the procedure from Section 4 can be adapted to the full CCP language. In Section 7 we discuss the use of the technique in Section 6 as a semi-decision procedure for observational equivalence. We also discuss the congruence issues concerning weak bisimilarity for CCP and CCP\+. Finally, in Section 8 we present our conclusions and future work.

2. Background

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, the concurrent constraint programming (CCP) and we show that partition refinement cannot be used for checking equivalence of concurrent constraint processes.

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 [19].

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 $\{B_1\} \ldots \{B_n\}$ to denote a partition consisting of (non-empty) blocks $B_1, \ldots, B_n$. 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.

2.1. Partition Refinement

We report the partition refinement algorithm [10] 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

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

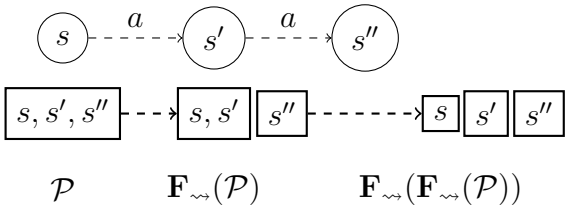

Figure 1: An example of the use of $\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$ from Equation 1

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 [19]).

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 $\mathcal{P}^n$.

2.2. 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 add to a *central shared store*. This information is represented as assertions traditionally referred to as *constraints*. Following [20, 21] 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. *(Constraint System) A* constraint system (cs) *is a complete algebraic lattice* $\mathbf{C} = (Con, Con_0, \sqsubseteq, \sqcup, true, false)$ *where* Con*, the set of constraints, is a partially ordered set w.r.t.* $\sqsubseteq$*,* 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.*

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

We now define the constraint system we use in our examples.

Example 1. *Let Var be a set of variables and ω be the set of natural numbers. A variable assignment is a function $\mu : Var \longrightarrow \omega$. We use $\mathcal{A}$ to denote the set of all assignments, $\mathcal{P}(\mathcal{A})$ to denote the powerset of $\mathcal{A}$, $\emptyset$ the empty set and $\cap$ the intersection of sets. Let us define the following constraint system: The set of constraints is $\mathcal{P}(\mathcal{A})$. We 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 $c \cap d$. We will often use a formula like $x < n$ to denote the corresponding constraint, i.e., the set of all assignments that map x to a number smaller than n.*

2.3. Syntax

We now recall the basic CCP process constructions. We are concerned with the verification of finite-state systems, thus we shall dispense with the recursion operator which is meant for describing infinite behavior. We shall also omit the local/hiding operator for the simplicity of the presentation (see [8, 9] for further details).

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

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

where $c \in Con_0$. Intuitively, $\mathbf{stop}$ represents termination, $\mathbf{tell}(c)$ adds the constraint (or partial information) c to the store. The addition is performed regardless the generation of inconsistent information. The process $\mathbf{ask}(c) \rightarrow P$ may execute P if c is entailed from 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.

Remark 2. *(CCP$\backslash$+). Henceforth, we use CCP$\backslash$+ to refer to the fragment of CCP without nondeterministic choice.*

2.4. Reduction Semantics

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 all configurations. We will use $Conf_{\texttt{CCP}\backslash+}$ for the CCP$\backslash$+ configurations.

R1	$\langle \mathbf{tell}(c), d\rangle \longrightarrow \langle \mathbf{stop}, d \sqcup c\rangle$	R2	$\dfrac{c \sqsubseteq d}{\langle \mathbf{ask}\ (c)\ \rightarrow\ P, d\rangle \longrightarrow \langle P, d\rangle}$
R3	$\dfrac{\langle P, d\rangle \longrightarrow \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \longrightarrow \langle P' \parallel Q, d'\rangle}$	R4	$\dfrac{\langle P, d\rangle \longrightarrow \langle P', d'\rangle}{\langle P\ +\ Q, d\rangle \longrightarrow \langle P', d'\rangle}$

Table 1: Reduction semantics for CCP (the symmetric rules for R3 and R4 are omitted).

The operational semantics of CCP is given by an *unlabeled* transition relation between 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 1. These rules are easily seen to realize the intuitions described in the syntax (Section 2.3).

In [8, 9], the authors introduced a *barbed semantics* for CCP. Barbed equivalences have been introduced in [22] for CCS, and have become a classical way to define the semantics of 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*, written $\gamma \downarrow_c$, iff $c \sqsubseteq d$. Similarly, γ satisfies a* weak barb c*, written $\gamma \Downarrow_c$, iff there exist γ' s.t. $\gamma \longrightarrow^* \gamma' \downarrow_c$.*

Example 2. *Let $\gamma = \langle \mathbf{ask}\ (x > 10)\ \rightarrow\ \mathbf{tell}(y < 42), x > 10\rangle$. We have $\gamma \downarrow_{x>5}$ since $(x > 5) \sqsubseteq (x > 10)$ and $\gamma \Downarrow_{y<42}$ since $\gamma \longrightarrow \langle \mathbf{tell}(y < 42), x > 10\rangle \longrightarrow \langle \mathbf{stop}, (x > 10) \sqcup (y < 42)\rangle \downarrow_{y<42}$.*

In this context, the equivalence proposed is the *saturated bisimilarity* [23, 24]. 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 is a saturated barbed bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$. We write $P \dot{\sim}_{sb} Q$ iff $\langle P, true\rangle \dot{\sim}_{sb} \langle Q, true\rangle$.*

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 ($\longrightarrow^*$).

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. $(\gamma_1, \gamma_2) \in \mathcal{R}$. We write $P \dot{\approx}_{sb} Q$ iff $\langle P, true\rangle \dot{\approx}_{sb} \langle Q, true\rangle$.

We now illustrate $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$ with the following two examples.

Example 3. *Take $T = \mathbf{tell}(true)$, $P = \mathbf{ask}\ (x < 7)\ \rightarrow\ T$ and $Q = \mathbf{ask}\ (x < 5)\ \rightarrow\ T$. One can check that $\langle P, true\rangle \not\dot{\sim}_{sb} \langle Q, true\rangle$, since $\langle P, x < 7\rangle \longrightarrow \langle T, x < 7\rangle$, while $\langle Q, x < 7\rangle \not\longrightarrow$. 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. $x < 7 \sqsubseteq e$, both the configurations evolve into $\langle T, e\rangle$, while for all e s.t. $x < 7 \not\sqsubseteq e$, both configurations cannot proceed. Since $x < 7 \sqsubseteq x < 5$, the behavior of Q is in a sense absorbed by the behavior of P.*

Example 4. *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. *The LTSs of* γ_1 *and* γ_2 *are the following:* $\gamma_1 \longrightarrow \langle \mathbf{stop}, true\rangle$ *and* $\gamma_2 \xrightarrow{c} \langle \mathbf{tell}(d), c\rangle \longrightarrow \langle \mathbf{stop}, c\rangle$. *It is now easy to see that the symmetric closure of the relation*

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

is a weak saturated barbed bisimulation as in Definition 4.

2.5. Labeled Semantics

In [8, 9] we have shown that $\dot{\approx}_{sb}$ is *fully abstract* with respect to the standard observational equivalence from [1]. Unfortunately, the quantification over all constraints in condition (iii) of Definition 3 and Definition 4 makes checking $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$ hard, since one should check infinitely many constraints. In order to avoid this problem we have introduced in [8, 9] a labeled transition semantics where labels are constraints.

In a transition of the form $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle$ the label $\alpha \in Con_0$ represents a *minimal* information (from the environment) that needs to be added to the store d to reduce from $\langle P, d\rangle$ to $\langle P', d'\rangle$, i.e., $\langle P, d \sqcup \alpha\rangle \longrightarrow \langle P', d'\rangle$. This approach is based on [25] which is related to the early work on symbolic semantics from [26].

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* $\rightsquigarrow$ *means* $\overset{true}{\rightsquigarrow}$. *Given a set of initial configurations* IS, $\mathtt{Config}_{\rightsquigarrow}(IS)$ *denotes the set* $\{\gamma' \mid \exists \gamma \in IS$ *s.t.* $\gamma \overset{\alpha_1}{\rightsquigarrow} \ldots \overset{\alpha_n}{\rightsquigarrow} \gamma'$ *for some* $n \geq 0\}$.

The LTS $(Conf, Con_0, \longrightarrow)$ is defined by the rules in Table 2. 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., $\alpha \in \min\{a \in Con_0 \,|\, c \sqsubseteq d \sqcup a\}$. The other rules are easily seen to realize the intuition given in Section 2.3. Figure 2 illustrates the LTS of our running example.

Given the LTS $(Conf, Con_0, \longrightarrow)$, one would like to exploit it for "efficiently characterizing" $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$. One first naive attempt would be to consider the standard notion of (weak) bisimilarity over $\longrightarrow$, but this would distinguish configurations which are in $\dot{\sim}_{sb}$ (and $\dot{\approx}_{sb}$), as illustrated by the following two examples.

$$\textsf{LR1}\ \langle \mathbf{tell}(c), d\rangle \xrightarrow{\mathit{true}} \langle \mathbf{stop}, d \sqcup c\rangle \qquad \textsf{LR2}\ \frac{\alpha \in \min\{a \in \mathit{Con}_0 \mid c \sqsubseteq d \sqcup a\}}{\langle \mathbf{ask}\ (c)\ \rightarrow\ P, d\rangle \xrightarrow{\alpha} \langle P, d \sqcup \alpha\rangle}$$

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

Table 2: Labeled semantics for CCP (the symmetric rules for LR3 and LR4 are omitted).

$T = \mathbf{tell}(\mathit{true})$ $\quad P = \mathbf{ask}\ (x < 7)\ \rightarrow\ T$ $\quad Q = \mathbf{ask}\ (x < 5)\ \rightarrow\ T$ $\quad R = \mathbf{ask}\ (z < 5)\ \rightarrow\ (P + Q)$
$T' = \mathbf{tell}(y = 1)$ $\quad S = \mathbf{ask}\ (z < 7)\ \rightarrow\ P$ $\quad Q' = \mathbf{ask}\ (x < 5)\ \rightarrow\ T'$ $\quad R' = \mathbf{ask}\ (z < 5)\ \rightarrow\ (P + Q')$

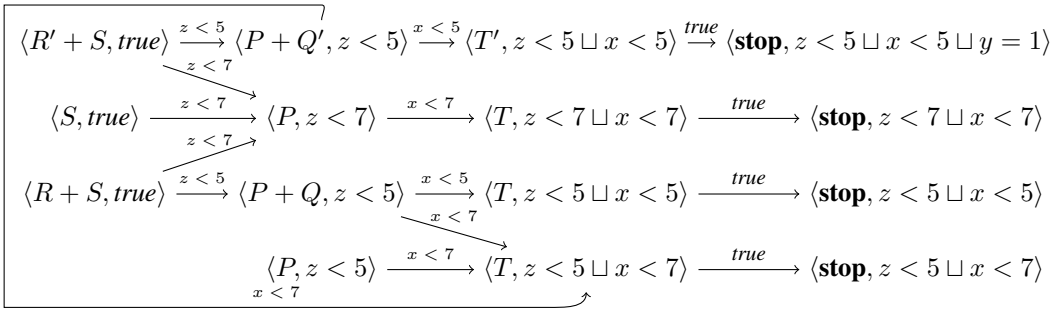

Figure 2: $\mathit{LTS}_{\longrightarrow}(\{\langle R' + S, \mathit{true}\rangle, \langle S, \mathit{true}\rangle, \langle R + S, \mathit{true}\rangle, \langle P, z < 5\rangle\})$.

Example 5. *In Example 3 we saw that* $\langle P + Q, true\rangle \dot{\sim}_{sb} \langle P, true\rangle$. *However,* $\langle P + Q, true\rangle \xrightarrow{x<5} \langle T, x < 5\rangle$, *while* $\langle P, true\rangle \not\xrightarrow{x<5}$.

Example 6. *In Example 4, we showed that* $\gamma_1 \dot{\approx}_{sb} \gamma_2$. *However,* $\gamma_2 \xrightarrow{c}$, *while* $\gamma_1 \not\xrightarrow{c}$

The examples above show that the ordinary notion of bisimilarity do not coincide with the intended semantics ($\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$). As a consequence, the standard partition refinement algorithm (Section 2.1) cannot be used for checking $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$. However, one can consider a variation of the bisimulation game, namely *irredundant bisimilarity* [15], which coincide with $\dot{\sim}_{sb}$ and, in the weak case [7], with $\dot{\approx}_{sb}$. This fact allowed us in [15] to define a variation of the partition refinement algorithm which we show in the next section.

First, we recall some results from [15] and [7], which are fundamental for the development of the paper.

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

Table 3: Weak semantics for CCP

Lemma 1. *([8, 9], [7]) (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 exists* α *and* b *s.t.* $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c''\rangle$ *where* $\alpha \sqcup b = a$ *and* $c'' \sqcup b = c'$.

The weak labeled transition system $(Conf, Con_0, \Longrightarrow)$ is defined by the rules in Table 3. This LTS is sound and complete, as $\longrightarrow$, and it can be used to decide $\dot{\approx}_{sb}$ as shown in [7].

Lemma 2. *([7]) (Soundness) If* $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c'\rangle$ *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 exists* α *and* b *s.t.* $\langle P, c\rangle \overset{\alpha}{\Longrightarrow} \langle P', c''\rangle$ *where* $\alpha \sqcup b = a$ *and* $c'' \sqcup b = c'$.

Note that we close $\longrightarrow$, not only w.r.t $true$ transitions (as similarly done in CCS, where τ intuitively corresponds to $true$), but w.r.t. *all* the transitions. This is needed to check $\dot{\approx}_{sb}$, because otherwise the above lemma would not hold.

Finally, the following lemma relates the labeled and weak semantics, i.e. $\longrightarrow$ and $\Longrightarrow$. It states that a single transition in $\Longrightarrow$ corresponds to a sequence of reductions ($\longrightarrow^*$).

Lemma 3. *([7])* $\gamma \longrightarrow^* \gamma'$ *iff* $\gamma \Longrightarrow \gamma'$.

3. Partition refinement for CCP

In this section we recall the partition refinement algorithm for CCP and how it can be used to decide observational equivalence.

3.1. Strong equivalence

In [15], we adapted the standard partition refinement procedure to check strong bisimilarity for CCP ($\dot{\sim}_{sb}$) by relying on [27] (which, in turn, was inspired by [28] presenting a similar procedure for the open π-calculus). As we did for the standard partition refinement, we also 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.

Now, since configurations satisfying different barbs are surely different, it can be safely started with a partition that equates all and only those states 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.

When splitting the above-mentioned partitions, 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 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 $t \succ t'$, iff $\alpha \sqsubset \beta$ and $c'' = c' \sqcup \beta$.*

The intuition is that the transition t dominates t' iff t requires less information from the environment than t' does (hence $\alpha \sqsubset \beta$), 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:*

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

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

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

One can check that $t_1 \succ t_2$ since $(x < 15) \sqsubset (x < 10)$ and also $(x < 10) = ((x < 15) \sqcup (x < 10))$.

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 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 $t \succ t''$, $lab(t'') = lab(t')$ 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 $\mathcal{R}$.*

To understand this definition better consider the following example.

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

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

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

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:

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

Finally, let $\mathcal{R} = \dot{\approx}_{sb}$ and take $t'' = (\gamma, b, \langle \mathbf{stop}, b\rangle)$. One can check that $t \succ_{\mathcal{R}} t'$ as in Definition 6. Firstly, $t \succ t''$ follows from $a \sqsubset b$. Secondly, we know $tar(t'')\mathcal{R}\ tar(t')$ from Example 4, i.e. $\langle \mathbf{stop}, b\rangle \dot{\approx}_{sb} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), b\rangle$ since $\langle \mathbf{stop}, true\rangle \dot{\approx}_{sb} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), true\rangle$.

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 states generated from the labeled semantics (Table 2) from the set of initial states should be included, i.e., $\mathtt{Config}_{\rightsquigarrow}(IS) \subseteq IS^{\star}_{\rightsquigarrow}$.

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 6 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 partition $\mathcal{P}^0$, also in the block of $\mathcal{P}^0$ where $\langle P_2, c_2\rangle$ is). See [15] for further details about the computation of $IS^{\star}_{\rightsquigarrow}$.

Finally, we shall describe how the refinement is done in the case CCP. 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.

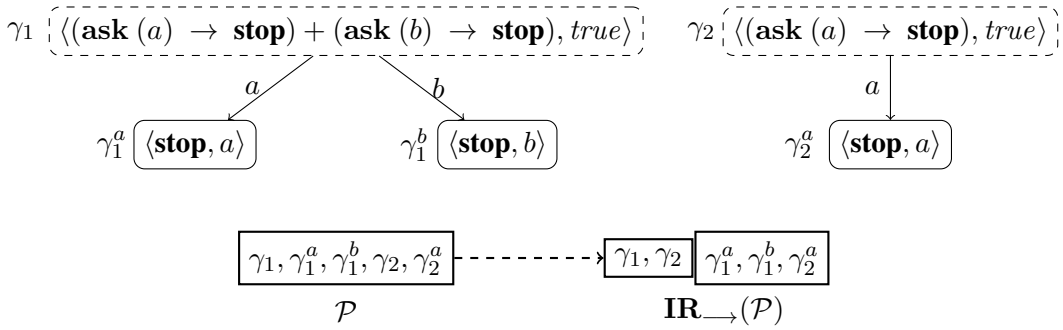

Figure 3: An example of the use of $\mathbf{IR}_{\longrightarrow}(\mathcal{P})$ as in Definition 7. Let $a \sqsubset b$, notice that γ_1 and γ_2 end up in the same block after the refinement since $\gamma_1 \xrightarrow{b} \gamma_1^b$ is a redundant transition w.r.t $\mathcal{P}$ hence it is not required that γ_2 matches it.

Definition 7. *(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 $\gamma_1'\ \mathcal{P}\ \gamma_2'$

See Figure 3 for an example of the use of $\mathbf{IR}_{\rightsquigarrow}(-)$.

Algorithm 2 $\texttt{pr-ccp}(IS, \rightsquigarrow)$

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 7
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

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

Theorem 1. *([15]) Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of $\texttt{pr-ccp}(IS, \longrightarrow)$ in Algorithm 2. Then*

- *$\gamma\ \mathcal{P}\ \gamma'$ iff $\gamma \dot{\sim}_{sb} \gamma'$.*

- $\texttt{pr-ccp}(IS, \longrightarrow)$ *may take exponential time in the size of* $\texttt{Config}_{\longrightarrow}(IS)$.

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

3.2. Weak equivalence

We can also use the above-mentioned algorithm to verify the weak version of saturated barbed bisimilarity ($\dot{\approx}_{sb}$). Recall that, as shown in [8, 9], in CCP\+ weak bisimilarity ($\dot{\approx}_{sb}$) coincides with the standard notion of CCP program (observational) equivalence.

Following [16] the reduction of the problem of deciding $\dot{\approx}_{sb}$ to the problem of deciding $\dot{\sim}_{sb}$ is obtained by adding some additional transitions, so called weak transitions, to the LTS. Given two configurations γ and γ', the first step is to build $G = LTS_{\longrightarrow}(IS)$ where $IS = \{\gamma, \gamma'\}$. Using G we then proceed to compute $G' = LTS_{\Longrightarrow}(IS)$, and finally we run Algorithm 2 adapted to G'. The adaptation consists in using weak barbs ($\Downarrow_c$) instead of barbs ($\downarrow_c$) for the initial partition $\mathcal{P}^0$ and using $\Longrightarrow$ as a parameter of the algorithm.

Algorithm 3 $\texttt{weak-pr-ccp}(IS, \rightsquigarrow)$

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 7
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

Using Algorithm 3 we can decide $\dot{\approx}_{sb}$ also with exponential time.

Theorem 2. *([7]) Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$ defined in Algorithm 3. Then*

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

As for the strong case, the exponential time is due to construction of the set $IS^{\star}_{\Longrightarrow}$ by $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$, whose size is exponential in $|\texttt{Config}_{\longrightarrow}(IS)|$. In the next section we shall address the issue of avoiding this exponential construction in the context of confluent CCP.

4. Using Partition refinement for checking observational equivalence in CCP\+

In the previous section, we presented a procedure to verify $\dot{\approx}_{sb}$ for CCP and we saw how this method takes exponential time (in the size of the LTS) to check whether two configurations are weakly bisimilar. In this section, we will explore what happens with such procedure when we restrict ourselves to CCP\+. We shall see that $\texttt{pr-ccp}(IS, \longrightarrow)$ may also be exponential time for inputs from the CCP\+ fragment.

Let us consider the following CCP\+ construction.

Example 9. *Let $n > 0$. We define $P^n = P^n_0$ with P^n_i, for $i \in \{0, \ldots, n-1\}$, given by:*

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

and $P^n_n = \mathbf{tell}(b_n)$. Furthermore, we assume that for all $i \in \{0, \ldots, n-1\}$ we have $a_i \sqsubseteq b_i$ 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 LTS for $\langle P^n, true\rangle$ is illustrated in Figure 4.

One can verify that by taking $IS = \{\langle P^n, true\rangle\}$ as in the example above, then the size of $IS^{\star}_{\longrightarrow}$ in Algorithm 2 grows exponentially with n, essentially because of the rule $(\mathsf{RD}^{IS}_{\longrightarrow})$.

Proposition 1. *Let $\gamma = \langle P^n, true\rangle$ as in Example 9 and take $IS = \{\gamma\}$. Let $\mathcal{P}$ be the output of $\texttt{pr-ccp}(IS, \longrightarrow)$ using Algorithm 2, then $\texttt{pr-ccp}(IS, \longrightarrow)$ takes at least exponential time in n.*

The main problem is that the procedure does not distinguish between summation-free processes and the normal CCP processes. Therefore, it is unable to exploit the underlying properties of CCP\+ and the algorithm will perform (in the worst-case) inherently the same as for the full CCP, as evidenced in the example above.

4.1. Properties of CCP\+

In this section we will state some features that (unlike the full CCP) this fragment possess. The first one we want to introduce is confluence. Intuitively, in CCP\+, if from a given configuration we have two possible reductions ($\longrightarrow$), then we are guaranteed that they will coincide at some point of the computation. Recall that $Conf_{\texttt{CCP}\backslash+}$ is the set of all CCP\+ configurations, i.e. configurations whose process is summation-free.

$$P^n = (\mathbf{ask}\ (a_0)\ \rightarrow\ (\mathbf{ask}\ (b_0)\ \rightarrow\ P_1^n))\ \|\ (\mathbf{ask}\ (b_0)\ \rightarrow\ \mathbf{stop})$$
$$LP_0^n = (\mathbf{ask}\ (b_0)\ \rightarrow\ P_1^n)\ \|\ (\mathbf{ask}\ (b_0)\ \rightarrow\ \mathbf{stop})$$
$$RP_0^n = (\mathbf{ask}\ (a_i)\ \rightarrow\ (\mathbf{ask}\ (b_i)\ \rightarrow\ P_{i+1}^n))\ \|\ \mathbf{stop})$$
$$LLP_0^n = P_1^n\ \|\ (\mathbf{ask}\ (b_0)\ \rightarrow\ \mathbf{stop})$$
$$LRP_0^n = (\mathbf{ask}\ (b_0)\ \rightarrow\ P_1^n)\ \|\ \mathbf{stop}$$

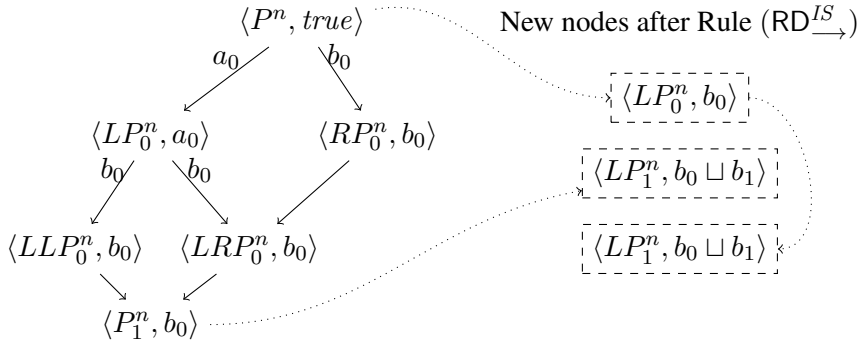

Figure 4: $LTS_{\longrightarrow}(IS)$ where $IS = \{\langle P^n, true\rangle\}$ as in Example 9. The configurations in the right part are generated by $(\mathsf{RD}^{IS}_{\longrightarrow})$ applied to the source nodes of the dotted arrows. Some transitions and **stop** processes were omitted for clarity.

Proposition 2. *([1]) Let $\gamma \in Conf_{\mathsf{CCP}\backslash+}$. If $\gamma \longrightarrow^* \gamma_1$ and $\gamma \longrightarrow^* \gamma_2$ then there exists γ' such that $\gamma_1 \longrightarrow^* \gamma'$ and $\gamma_2 \longrightarrow^* \gamma'$.*

Before discussing the second property, we need to introduce some notation. We shall call *derivatives* (of γ) the successors reached via (zero or more) reductions ($\longrightarrow^*$) starting from a given configuration γ.

Definition 8. *(Derivatives) The derivatives of a configuration γ, written $\mathsf{Deriv}(\gamma)$, are defined as $\mathsf{Deriv}(\gamma) = \{\gamma' \mid \gamma \longrightarrow^* \gamma'\}$.*

Using this notation, we can now state another property of CCP\+. A CCP\+ configuration is weakly bisimilar to all its derivatives.

Lemma 4. *Let $\gamma \in Conf_{\mathsf{CCP}\backslash+}$. For all $\gamma' \in \mathsf{Deriv}(\gamma)$ we have $\gamma \dot{\approx}_{sb} \gamma'$.*

Proof. Let $\mathcal{R} = \{(\gamma_1, \gamma_2) \mid \exists \gamma_3$ s.t. $\gamma_1 \longrightarrow^* \gamma_3$ and $\gamma_2 \longrightarrow^* \gamma_3\}$, we prove that $\mathcal{R}$ is a weak saturated barbed bisimulation. Let (γ_1, γ_2) be any pair of configurations in $\mathcal{R}$.

(i) If $\gamma_1 \Downarrow_e$ then by definition $\gamma_1 \longrightarrow^* \gamma_1' \downarrow_e$. By confluence (Proposition 2) $\gamma_1' \longrightarrow^* \gamma_3$ and thus $\gamma_3 \downarrow_e$ (since constraints can only be added). Since $\gamma_2 \longrightarrow^* \gamma_3 \downarrow_e$ we conclude that $\gamma_2 \Downarrow_e$.
(ii) If $\gamma_1 \longrightarrow^* \gamma_1'$, then by confluence $\gamma_1' \longrightarrow^* \gamma_3$ and therefore $(\gamma_1', \gamma_2) \in \mathcal{R}$.
(iii) Finally, let $\gamma_1 = \langle P_1, c_1 \rangle$ and $\gamma_2 = \langle P_2, c_2 \rangle$. If $\langle P_1, c_1 \rangle \longrightarrow^* \langle P_3, c_3 \rangle$ and $\langle P_2, c_2 \rangle \longrightarrow^* \langle P_3, c_3 \rangle$, then $\langle P_1, c_1 \sqcup e \rangle \longrightarrow^* \langle P_3, c_3 \sqcup e \rangle$ and $\langle P_2, c_2 \sqcup e \rangle \longrightarrow^* \langle P_3, c_3 \sqcup e \rangle$ and thus $(\langle P_1, c_1 \sqcup e \rangle, \langle P_2, c_2 \sqcup e \rangle) \in \mathcal{R}$. □

The proof above relies on the intrinsic confluent nature of CCP\+ (Proposition 2) and this lemma will be central for the results we will present next. In the next section we shall take advantage of these properties to check $\dot{\approx}_{sb}$ for CCP\+ configurations.

4.2. Optimizations to partition refinement for CCP\+

We presented how the partition refinement for CCP performs for CCP\+ as well as some properties of the configurations of this fragment. In this section, using such features, we shall show that the complexity of `weak-pr-ccp`$(IS, \Longrightarrow)$ can be improved, thus we can check $\dot{\approx}_{sb}$ in a more efficient manner.

Due to the nature of CCP\+, determining which are the redundant transitions w.r.t. $\dot{\approx}_{sb}$ (Definition 6) becomes an easier task. As we explained in Section 3.1, the purpose of rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ from Table 4 is to add some configurations to $IS^{\star}_{\rightsquigarrow}$ that will be used to check redundancy at each iteration of Algorithm 2. In CCP\+ these additional configurations are not necessary. But before we arrive to this let us introduce some definitions first.

Definition 9. *We say that γ goes with α to γ' with a* maximal weak transition*, written $\gamma \stackrel{\alpha}{\Longrightarrow}_{\max} \gamma'$, iff $\gamma \stackrel{\alpha}{\Longrightarrow} \gamma' \not\longrightarrow$.*

The definition above reflects the fact that when $\gamma \stackrel{\alpha}{\Longrightarrow}_{\max} \gamma'$ then γ' has no more information to deduce without the aid of the environment, namely no further reduction ($\longrightarrow$) is possible. As $\Longrightarrow$, the maximal weak transition relation $\Longrightarrow_{\max}$ is sound and complete.

Lemma 5. *(Soundness) If $\langle P, c \rangle \stackrel{\alpha}{\Longrightarrow}_{\max} \langle P', c' \rangle$ then $\langle P, c \sqcup \alpha \rangle \Longrightarrow_{\max} \langle P', c' \rangle$. (Completeness) If $\langle P, c \sqcup a \rangle \Longrightarrow_{\max} \langle P', c' \rangle$ then there exists α and b such that $\langle P, c \rangle \stackrel{\alpha}{\Longrightarrow}_{\max} \langle P', c'' \rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$.*

Proof. Follows from the correctness of $\Longrightarrow$ (Lemma 2) and from the fact that $LTS_{\longrightarrow}(\{\langle P, c \rangle\})$ is finite. □

As one would expect, $\Longrightarrow_{\max}$ can also be used to compute $\dot{\approx}_{sb}$ and the complexity of the procedure is equivalent to the case of $\Longrightarrow$ (Theorem 2).

Theorem 3. *([7]) Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$, let $\mathcal{P}$ be the output* `weak-pr-ccp`$(IS, \Longrightarrow_{\max})$ *using Algorithm 3. Then*

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

Proof. Follows from the correctness of $\Longrightarrow_{\max}$ (Lemma 5); Corollary 1 and Theorem 3 from [7] and Theorem 2. □

Nevertheless, in CCP\+, the maximal weak transitions $\Longrightarrow_{\max}$ satisfy a particular property that allow us to erase the redundant transitions w.r.t. $\dot{\approx}_{sb}$ before computing $\dot{\approx}_{sb}$ itself.

Proposition 3. *Let $\gamma = \langle P, c\rangle \in Conf_{\texttt{CCP}\backslash+}$. Let $t_1 = \gamma \overset{\alpha}{\Longrightarrow}_{\max} \langle P_1, c_1\rangle$ and $t_2 = \gamma \overset{\beta}{\Longrightarrow}_{\max} \langle P_2, c_2\rangle$. We have that $\alpha \sqsubset \beta$ and $\langle P_1, c_1 \sqcup \beta\rangle \longrightarrow^* \langle P', c_2\rangle \not\longrightarrow$ iff $t_1 \succ_{\dot{\approx}_{sb}} t_2$.*

Proof. **(⇒)** By soundness on t_1 we have $\langle P, c \sqcup \alpha\rangle \Longrightarrow_{\max} \langle P_1, c_1\rangle$ then by definition $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P_1, c_1\rangle$ now by monotonicity $\langle P, c \sqcup \beta\rangle \Longrightarrow \langle P_1, c_1 \sqcup \beta\rangle$ and then $\langle P, c \sqcup \beta\rangle \longrightarrow^* \langle P_1, c_1 \sqcup \beta\rangle$ then by Lemma 4 $\langle P, c \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_1, c_1 \sqcup \beta\rangle$. Using a similar reasoning on t_2 we can conclude that $\langle P, c \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$ and by transitivity $\langle P_1, c_1 \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$. Finally take $t' = (\gamma, \beta, \langle P_1, c_1 \sqcup \beta\rangle)$, hence we can conclude that $t_1 \succ_{\dot{\approx}_{sb}} t_2$ since $t_1 \succ t'$ and $\langle P_1, c_1 \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$.

(⇐) Assume that $t_1 \succ_{\dot{\approx}_{sb}} t_2$ then there exists $t' = (\gamma, \beta, \langle P_1, c'\rangle)$ such that $t_1 \succ t'$ and $\langle P_1, c'\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$. By $t_1 \succ t'$ we know that $\alpha \sqsubset \beta$ and $c' = c_1 \sqcup \beta$. Now since $\langle P_2, c_2\rangle \not\longrightarrow$ by definition of $\Longrightarrow_{\max}$, therefore by condition (i) of $\dot{\approx}_{sb}$ we have $c' \sqsubseteq c_2$. Moreover, $\langle P_1, c'\rangle \longrightarrow^* \langle P', c_3\rangle$ where $c_2 \sqsubseteq c_3$. By contradiction let $c_2 \neq c_3$ then $c_2 \sqsubset c_3$, thus there is e s.t. $\langle P_1, c'\rangle \Downarrow_e$ but since $\langle P_2, c_2\rangle \not\longrightarrow$ then $\langle P_2, c_2\rangle \not\Downarrow_e$ and so $\langle P_1, c'\rangle \not\dot{\approx}_{sb} \langle P_2, c_2\rangle$, an absurd. Thus $c_3 = c_2$ hence $\langle P_1, c'\rangle \longrightarrow^* \langle P', c_2\rangle \not\longrightarrow$. □

Using this property we can define a new procedure for deciding $\dot{\approx}_{sb}$ that does not use Rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ since redundancy can be checked and erased using Proposition 3 (Algorithm 4, Step 2).

In other words, we first need to build the LTS of reachable states using $\Longrightarrow_{\max}$. Then for each pair of transitions with the same source we can check whether

Proposition 3 holds, if it is the case then the redundant transition is removed. Finally we apply the standard partition refinement on the LTS resulting from the previous step.

Algorithm 4 $\texttt{weak-pr-dccp}(IS)$

Initialization

1. Compute $G = LTS_{\Longrightarrow_{\max}}(IS)$ using the rules $(\mathsf{IS}^{IS}_{\Longrightarrow_{\max}})$ and $(\mathsf{RS}^{IS}_{\Longrightarrow_{\max}})$,
2. $G' = \texttt{remRed}(G)$ where the graph $\texttt{remRed}(G)$ results from removing from G the redundant transitions w.r.t. $\dot{\approx}_{sb}$,
3. $\mathcal{P}^0 = \{B_1\} \ldots \{B_m\}$ is a partition of $\texttt{V}(G')$ where γ and γ' are in B_i iff they satisfy the same weak barbs ($\Downarrow_e$),

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

The key idea is that in order to compute $\dot{\approx}_{sb}$, with the redundancy removed, it suffices to refine the partitions using $\mathbf{F}_{\Longrightarrow_{\max}}(\mathcal{P})$ (defined by Equation 1) instead of $\mathbf{IR}_{\Longrightarrow_{\max}}(\mathcal{P})$. Algorithm 4 can be used to decide $\dot{\approx}_{sb}$ for configurations in $Conf_{\texttt{CCP}\setminus+}$ with polynomial time.

Theorem 4. *Let γ and γ' be two CCP\+ configurations. Let $IS = \{\gamma, \gamma'\}$, let $\mathcal{P}$ be the output of $\texttt{weak-pr-dccp}(IS)$ in Algorithm 4 and let $N = |\texttt{Config}_{\longrightarrow}(IS)|$. Then*

- *$\gamma \; \mathcal{P} \; \gamma'$ iff $\gamma \dot{\approx}_{sb} \gamma'$.*
- *$\texttt{weak-pr-dccp}(IS)$ takes $O(N^3)$ time and uses $O(N^2)$ space.*

Proof. The first item follows from the Theorem 2 and Proposition 3. As for the second item:
(Step 1) $G = LTS_{\Longrightarrow_{\max}}(IS)$ takes $O(N^2)$ time and space since $\Longrightarrow_{\max}$ will add, at most, a transition from each element in $\texttt{V}(G)$ to every other configuration in $\texttt{V}(G)$ and $|\texttt{V}(G)| = |\texttt{Config}_{\longrightarrow}(IS)| = N$.
(Step 2) Each node in $\texttt{V}(G)$ has at most $N-1$ outgoing transitions, then $G' = \texttt{remRed}(G)$ takes $O((N-1) * (N-1)) = O(N^2)$ per node, thus this step takes $O(N^2 * N) = O(N^3)$ time.
(Step 3) $\mathcal{P}^0$ can be created in $O(N^2)$ by definition of $\Longrightarrow_{\max}$.

(Iteration) Using the procedure from Tarjan et al. [29], this step takes $O(|E| \log |V|)$ time and uses $O(|E|)$ space. Therefore, since $|\mathtt{V}(G)| = N$ and $|\mathtt{E}(G)| = O(N^2)$, hence we have $O(N^2 \log N)$ and $O(N^2)$ space.
We can conclude that $\mathtt{weak\text{-}pr\text{-}dccp}(IS)$ takes $O(N^3)$ time and uses $O(N^2)$ space. □

Thanks to Proposition 3, by removing redundant transitions, we can solve the problem of checking bisimilarity for CCP\+ with the standard solutions for checking bisimilarity. In Algorithm 4, we have used the "classical" partition refinement, but different, more effective solutions, are possible. For instance, executing the algorithm in [13] (after having removed all the redundant transitions) would require at most $O(|E| + |V|)$ steps. Note however that, due to the closure needed for weak transitions (Table 3), $|E|$ is usually quadratic w.r.t. the number of states $|V|$. In the following section, we introduce a novel procedure which avoids such expensive closure.

5. Using the compact input-output sets for verifying observational equivalence in CCP\+

In the previous section we improved the CCP exponential-time decision procedure for $\dot{\approx}_{sb}$ to obtain a polynomial-time procedure for the special case of the summation-free fragment CCP\+. (Recall that in CCP\+, the relation $\dot{\approx}_{sb}$ coincides with the standard notion of observational equivalence.)

In this section, we will present an alternative approach for verifying observational equivalence for CCP\+ that improves on the time and space complexity of Algorithm 4.

Roughly speaking our approach consists in reducing the problem of whether two given CCP\+ configurations γ, γ' are in $\dot{\approx}_{sb}$ to the problem of whether γ and γ' have the same minimal finite representation of the set of weak barbs they satisfy in every possible context.

5.1. Weak bisimilarity and barb equivalence

First we will show that, in CCP\+, we can give characterization of $\dot{\approx}_{sb}$ in terms of the simpler notion of weak-barb equivalence defined below. Intuitively, two configurations are saturated weakly bisimilar if and only if for *every* possible augmentation of their stores, the resulting configurations satisfy the same weak barbs. More precisely,

Definition 10. *(Barb equivalence)* $\langle P, c\rangle$ *and* $\langle Q, d\rangle$ *are (weak) barbed equivalent, written* $\langle P, c\rangle \sim_{wb} \langle Q, d\rangle$*, iff*

$$\forall e, \alpha \in Con_0.\ \langle P, c \sqcup e\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d \sqcup e\rangle \Downarrow_\alpha$$

Let us give an example.

Example 10. *Let* $P = \mathbf{tell}(true)$ *and* $Q = \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d)$*. We can show that* $\langle P, true\rangle \sim_{wb} \langle Q, true\rangle$ *when* $d \sqsubseteq c$ *(as in Example 4). One can check that for all* c' *we have* $\langle P, c'\rangle \Downarrow_{c'}$ *and* $\langle Q, c'\rangle \Downarrow_{c'}$*. Notice that whenever* c *is entailed then* $\mathbf{tell}(d)$ *does not add any more information since* $d \sqsubseteq c$*.*

The full characterization of $\dot{\approx}_{sb}$ in terms of weak-barbed equivalence is given next. Notice that the following theorem uses Lemma 4 which itself depends on the confluent nature of CCP\+.

Theorem 5. *Let* $\langle P, c\rangle$ *and* $\langle Q, d\rangle$ *be CCP\+ configurations.* $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$ *iff* $\langle P, c\rangle \sim_{wb} \langle Q, d\rangle$

Proof. **(⇒)** Assume that $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$ then by condition (i) of $\dot{\approx}_{sb}$ (Definition 3) we have $\forall \alpha \in Con_0. \langle P, c\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d\rangle \Downarrow_\alpha$, hence in combination with condition (iii) we can conclude $\langle P, c \sqcup e\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d \sqcup e\rangle \Downarrow_\alpha$.

(⇐) Let $\mathcal{R} = \{(\langle P, c\rangle, \langle Q, d\rangle) \mid \forall e, \alpha \in Con_0.\ \langle P, c \sqcup e\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d \sqcup e\rangle \Downarrow_\alpha\}$, we prove that $\mathcal{R}$ is a weak saturated barbed bisimulation:
(i) Take $e = true$ then $\forall \alpha \in Con_0. \langle P, c\rangle \Downarrow_\alpha \Leftrightarrow \langle Q, d\rangle \Downarrow_\alpha$.
(ii) Assume that $\langle P, c\rangle \longrightarrow^* \langle P', c'\rangle$, by Lemma 4 $\langle P, c\rangle \dot{\approx}_{sb} \langle P', c'\rangle$ hence by (⇒) we can conclude that $\langle P', c'\rangle \mathcal{R} \langle Q, d\rangle$.
(iii) Assume $\langle P, c\rangle \mathcal{R} \langle Q, d\rangle$ then for all e' we have $\langle P, c \sqcup e'\rangle \mathcal{R} \langle Q, d \sqcup e'\rangle$ just by taking $e = e'$. □

We shall show a compact representation of the set of weak barbs of a configuration under any possible context. First we introduce some convenient notation for this purpose. The set $[\![\langle P, c\rangle]\!]$ will contain pairs of the form (α, e).

Definition 11. *(Input-Output set) The* input-output *set of a given configuration* $\langle P, c\rangle$ *is defined as follows:*

$$[\![\langle P, c\rangle]\!] \stackrel{\text{def}}{=} \{(\alpha, e) \mid \langle P, c \sqcup \alpha\rangle \Downarrow_e\}$$

Let us give an example.

Example 11. *Let* $\gamma = \langle \mathbf{ask}\ a\ \rightarrow\ \mathbf{tell}(b), true\rangle$ *where* $b \not\sqsubseteq a$.

$$[\![\gamma]\!] = \{(\alpha,\alpha)|\alpha \sqsubset a \text{ or } a \not\sqsubseteq \alpha\} \cup \{(\beta, \beta \sqcup b)|a \sqsubseteq \beta\}$$

Intuitively, each pair $(\alpha, e) \in [\![\langle P, c\rangle]\!]$ denotes a stimulus-response, or input-output, interaction of $\gamma = \langle P, c\rangle$: If the environment adds α to the store of γ, the resulting configuration $\langle P, c \sqcup \alpha\rangle$ may evolve, without any further interaction with the environment, into a configuration whose store entails e. In other words $\langle P, c \sqcup \alpha\rangle\ \Downarrow_e$. We can think of e as piece of information that $\langle P, c \sqcup \alpha\rangle$ may produce.

The following corollary is an immediate consequence of the definitions.

Corollary 1. $[\![\langle P, c\rangle]\!] = [\![\langle Q, d\rangle]\!]$ *iff* $\langle P, c\rangle \sim_{wb} \langle Q, d\rangle$

Proof. ($\Rightarrow$) Assume $(\beta, a) \in [\![\langle P, c\rangle]\!]$ then $\langle P, c \sqcup \beta\rangle\ \Downarrow_a$. Hence, we can take $e = \beta$ and by hypothesis $\langle Q, d \sqcup \beta\rangle\ \Downarrow_a$ therefore $(\beta, a) \in [\![\langle Q, d\rangle]\!]$.

($\Leftarrow$) Assume that $\langle P, c \sqcup \beta\rangle\ \Downarrow_a$ for some β and a, then $(\beta, a) \in [\![\langle P, c\rangle]\!]$ and by hypothesis $(\beta, a) \in [\![\langle Q, d\rangle]\!]$ therefore by definition $\langle Q, d \sqcup \beta\rangle\ \Downarrow_a$.

□

We now introduce the notion of relevant input-output pair.

Definition 12. *(Relevant Pair) Let* (α, e) *and* (β, e') *be elements from* $Con_0 \times Con_0$. *We say that* (α, e) *is more* relevant *than* (β, e'), *written* $(\alpha, e) \succeq (\beta, e')$, *iff* $\alpha \sqsubseteq \beta$ *and* $e' \sqsubseteq (e \sqcup \beta)$. *Similarly, given* $p = (\beta, e')$ *s.t.* $p \in \mathcal{S}$, *we say that the pair* p *is* irrelevant *in* $\mathcal{S}$ *if there is a pair* $(\alpha, e) \in \mathcal{S}$ *more relevant than* p, *else* p *is said to be* relevant *in* $\mathcal{S}$.

Let us illustrate this with an example.

Example 12. *Let* $\mathcal{S} = \{(true, true), (\alpha, \alpha), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$ *where* $\alpha, \beta, c \in Con_0$ *are not related to each other. Notice that* $(true, true) \succeq (\alpha, \alpha)$ *however* $(true, true) \not\succeq (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)$. *This means that* $(true, true)$ *and* $(\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)$ *are relevant in* $\mathcal{S}$ *and* (α, α) *is irrelevant in* $\mathcal{S}$.

Recall the stimulus-response intuition given above. In other words, the pair (β, e') is *irrelevant* in a given input-output set if there exists another pair (α, e) in the set that represents the need for less stimulus from the environment, hence the condition $\alpha \sqsubseteq \beta$, to produce at least as much information, with the possible exception of information that β may entail but α does not. Hence $e' \sqsubseteq e \sqcup \beta$.

We now list two important properties of $\succeq$ that will be useful later on. The set $[\![\langle P, c\rangle]\!]$ is closed w.r.t. $\succeq$.

Proposition 4. *Let* $(\alpha, e) \in [\![\langle P, c\rangle]\!]$. *If* $(\alpha, e) \succeq (\beta, e')$ *then* $(\beta, e') \in [\![\langle P, c\rangle]\!]$.

Proof. By definition $\langle P, c \sqcup \alpha\rangle \Downarrow_e$ then by monotonicity $\langle P, c \sqcup \beta\rangle \Downarrow_{e \sqcup \beta}$ so $\langle P, c \sqcup \beta\rangle \Downarrow_{e'}$ since $e' \sqsubseteq (e \sqcup \beta)$, therefore $(\beta, e') \in [\![\langle P, c\rangle]\!]$. □

Moreover, the relation $\succeq$ is well-founded. More precisely,

Proposition 5. *There is no infinite* strictly *descending chain* $p_1 \succ p_2 \succ \ldots$.

Proof. Follows from the well-foundedness of $\sqsubseteq$ (Remark 1). □

5.2. A canonical representation of CCP\+ configurations

Clearly $[\![\langle P, c\rangle]\!]$ may be infinite due potential existence of infinitely many arbitrary stimuli (inputs). By using the labeled transition semantics (Table 2) we shall show that we do not need consider arbitrary inputs but only the minimal ones. Recall that in $\gamma \xrightarrow{\alpha} \gamma'$ the label α represents the minimal information needed to evolve from γ to γ'.

Definition 13. *The* label-based input-output set *of a configuration* $\langle P, c\rangle$*, denoted as* $\mathcal{M}(\langle P, c\rangle)$*, is inductively defined as follows:*

$$\{(true, c)\} \cup \bigcup_{\langle P,c\rangle \xrightarrow{\alpha} \langle P',c'\rangle} (\{(\alpha, c')\} \cup (\alpha \otimes \mathcal{M}(\langle P', c'\rangle)))$$

where $\otimes : Con_0 \times 2^{Con_0 \times Con_0} \longrightarrow 2^{Con_0 \times Con_0}$ *is defined as*

$$\alpha \otimes \mathcal{S} \stackrel{\text{def}}{=} \{(\alpha \sqcup \beta, e) \mid (\beta, e) \in \mathcal{S})\}$$

Let us illustrate this definition with an example.

Example 13. *Let* $\gamma = \langle \mathbf{ask}\ (\alpha) \rightarrow (\mathbf{ask}\ (\beta) \rightarrow \mathbf{tell}(c)), true\rangle$ *and* $\gamma' = \langle \mathbf{ask}\ (\alpha \sqcup \beta) \rightarrow \mathbf{tell}(c), true\rangle$ *where* $\alpha, \beta, c \in Con_0$ *are not related to each other. Let us assume that* α *and* β *are the minimal elements that allow* γ *and* γ' *to proceed. Their corresponding label-based input-output sets are as follows:*

$$\mathcal{M}(\gamma) = \{(true, true), (\alpha, \alpha), (\alpha \sqcup \beta, \alpha \sqcup \beta), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$$

$$\mathcal{M}(\gamma') = \{(true, true), (\alpha \sqcup \beta, \alpha \sqcup \beta), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$$

Nevertheless, labeled-based input-output sets do not give us a fully-abstract representation of the input-output sets because of the existence of irrelevant pairs. By excluding these pairs we obtain a compact and fully-abstract representation of input-output sets.

Definition 14. *(Compact input-output set) The* compact *input-output set of a configuration $\langle P, c\rangle$ is defined as follows:*

$$\mathcal{M}^C(\langle P, c\rangle) \stackrel{\texttt{def}}{=} \{(\alpha, e) \mid (\alpha, e) \in \mathcal{M}(\langle P, c\rangle) \textit{ and } (\alpha, e) \textit{ is relevant in } \mathcal{M}(\langle P, c\rangle)\}$$

Let us give an example.

Example 14. *Let γ and γ' as in Example 13. Using the same reasoning as in Example 12 one can check that:*

$$\mathcal{M}^C(\gamma) = \mathcal{M}^C(\gamma') = \{(true, true), (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)\}$$

We shall now show the full-abstraction of the compact input-output sets. We need the following lemmata. First, compact sets are closed under weak transitions ($\Longrightarrow$). More precisely:

Proposition 6. *If $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c'\rangle$ then $(\alpha, c') \in \mathcal{M}(\langle P, c\rangle)$.*

Proof. By induction on the depth of the inference of $\langle P, c\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', c'\rangle$. Consider the rules R-Tau, R-Label and R-Add from Table 5.

- Using Rule R-Tau we have $\langle P, c\rangle \Longrightarrow \langle P, c\rangle$ and $(true, c) \in \mathcal{M}(\langle P, c\rangle)$ by definition.
- Using Rule R-Label we have $\langle P, c\rangle \stackrel{\alpha}{\longrightarrow} \langle P', c'\rangle$ and $(\alpha, c') \in \mathcal{M}(\langle P, c\rangle)$ by definition.
- Using Rule R-Add we have $\langle P, c\rangle \stackrel{\alpha''}{\Longrightarrow} \langle P'', c''\rangle \stackrel{\alpha'}{\Longrightarrow} \langle P', c'\rangle$ where $\alpha' \sqcup \alpha'' = \alpha$. Then by induction hypothesis $(\alpha'', c'') \in \mathcal{M}(\langle P, c\rangle)$ and $(\alpha', c') \in \mathcal{M}(\langle P'', c''\rangle)$, hence by definition of $\mathcal{M}(\langle P, c\rangle)$ we have $(\alpha' \sqcup \alpha'', c') \in \mathcal{M}(\langle P, c\rangle)$ so $(\alpha, c') \in \mathcal{M}(\langle P, c\rangle)$.

□

The following proposition states that whenever a pair (α, e) is in $\mathcal{M}(\langle P, c\rangle)$, it means that e can be reached from $\langle P, c \sqcup \alpha\rangle$ without aid of the environment.

Proposition 7. *If $(\alpha, e) \in \mathcal{M}(\langle P, c\rangle)$ then $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', e\rangle$*

Proof. By definition of $\mathcal{M}(\langle P, c\rangle)$, since $(\alpha, e) \in \mathcal{M}(\langle P, c\rangle)$ then there exist $\alpha_1, \ldots, \alpha_n$ such that $\alpha = \bigsqcup_{i=1}^n \alpha_n$ and $\langle P, c\rangle \stackrel{\alpha_1}{\longrightarrow} \ldots \stackrel{\alpha_n}{\longrightarrow} \langle P', e\rangle$. Hence by soundness on each transition $\langle P, c \sqcup \bigsqcup_{i=1}^n \alpha_n\rangle = \langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', e\rangle$. □

We can now prove our main result, given two configurations $\langle P, c\rangle$ and $\langle Q, d\rangle$, they are observationally equivalent if and only if their compact input-output sets are identical. We split the proof in the following two lemmata.

Lemma 6. *If* $\mathcal{M}^C(\langle P, c\rangle) = \mathcal{M}^C(\langle Q, d\rangle)$ *then* $[\![\langle P, c\rangle]\!] = [\![\langle Q, d\rangle]\!]$

Proof. Let us assume that $(\alpha, \beta) \in [\![\langle P, c\rangle]\!]$ then by definition $\langle P, c \sqcup \alpha\rangle \Downarrow_\beta$ hence there exists P' and β' such that $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', \beta'\rangle$ and $\beta \sqsubseteq \beta'$. By Lemma 3 we have $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P', \beta'\rangle$, then by completeness of $\Longrightarrow$ (Lemma 2) there exist α', b s.t. $\langle P, c\rangle \stackrel{\alpha'}{\Longrightarrow} \langle P', c'\rangle$ where $\alpha' \sqcup b = \alpha$ and $c' \sqcup b = \beta'$ **(1)**. Now by Proposition 6 we know $(\alpha', c') \in \mathcal{M}(\langle P, c\rangle)$, then since $\succeq$ is well-founded (Proposition 5) there is (α'', c'') that is relevant in $\mathcal{M}(\langle P, c\rangle)$ (then it belongs to $\mathcal{M}^C(\langle P, c\rangle)$) such that $(\alpha'', c'') \succeq (\alpha', c')$, namely $\alpha'' \sqsubseteq \alpha'$ (or equivalently $\exists x.(\alpha'' \sqcup x) = \alpha'$ **(2)**) and $c' \sqsubseteq (c'' \sqcup \alpha')$. Given that $(\alpha'', c'') \in \mathcal{M}^C(\langle P, c\rangle)$ then by hypothesis $(\alpha'', c'') \in \mathcal{M}^C(\langle Q, d\rangle)$, this means also that $(\alpha'', c'') \in \mathcal{M}(\langle Q, d\rangle)$ and by Proposition 7 we know that $\langle Q, d \sqcup \alpha''\rangle \longrightarrow^* \langle Q', c''\rangle$. By monotonicity we have the following transition $\langle Q, d \sqcup \alpha'' \sqcup x \sqcup b\rangle \longrightarrow^* \langle Q', c'' \sqcup x \sqcup b\rangle$, now notice that from **(1)** and **(2)** we have $(d \sqcup \alpha'' \sqcup x \sqcup b) = (d \sqcup \alpha' \sqcup b) = (d \sqcup \alpha)$ then $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', c'' \sqcup x \sqcup b\rangle$. Finally, we have to prove that $\beta \sqsubseteq (c'' \sqcup x \sqcup b)$ to conclude that $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$, for that purpose, recall that $\beta \sqsubseteq \beta' = (c' \sqcup b) \sqsubseteq (c'' \sqcup \alpha' \sqcup b)$ and since $(c'' \sqcup \alpha'' \sqcup x \sqcup b) = (c'' \sqcup x \sqcup b)$ then $\beta \sqsubseteq (c'' \sqcup x \sqcup b)$ and so $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$. □

Lemma 7. *If* $[\![\langle P, c\rangle]\!] = [\![\langle Q, d\rangle]\!]$ *then* $\mathcal{M}^C(\langle P, c\rangle) = \mathcal{M}^C(\langle Q, d\rangle)$

Proof. Assume that $(\alpha, \beta) \in \mathcal{M}^C(\langle P, c\rangle)$ our goal is to prove that $(\alpha, \beta) \in \mathcal{M}^C(\langle Q, d\rangle)$. By definition (α, β) is relevant in $\mathcal{M}(\langle P, c\rangle)$, moreover, by Proposition 7 we have $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', \beta\rangle$ then by definition $(\alpha, \beta) \in [\![\langle P, c\rangle]\!]$ and by hypothesis $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$. This means that $\langle Q, d \sqcup \alpha\rangle \Downarrow_\beta$ then there exists Q', d' s.t. $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\beta \sqsubseteq d'$. By Lemma 3 we have $\langle Q, d \sqcup \alpha\rangle \Longrightarrow \langle Q', d'\rangle$, now by completeness of $\Longrightarrow$ (Lemma 2) there exist α', b s.t. $\langle Q, d\rangle \stackrel{\alpha'}{\Longrightarrow} \langle Q', d''\rangle$ where $(\alpha' \sqcup b) = \alpha$ and $(d'' \sqcup b) = d'$. Now let us assume by means of contradiction that $\alpha' \neq \alpha$. By soundness of $\Longrightarrow$ (Lemma 2) we have $\langle Q, d \sqcup \alpha'\rangle \Longrightarrow \langle Q', d''\rangle$ then by Lemma 3 we get $\langle Q, d \sqcup \alpha'\rangle \longrightarrow^* \langle Q', d''\rangle$ hence $(\alpha', d'') \in [\![\langle Q, d\rangle]\!]$. By hypothesis then $(\alpha', d'') \in [\![\langle P, c\rangle]\!]$, now this means that $\langle P, c \sqcup \alpha'\rangle \longrightarrow^* \langle P'', e\rangle$ where $d'' \sqsubseteq e$ (equivalently $\exists z.(d'' \sqcup z) = e$). By Lemma 3 we get that $\langle P, c \sqcup \alpha'\rangle \Longrightarrow \langle P'', e\rangle$ and by completeness there exist x, b' s.t. $\langle P, c\rangle \stackrel{x}{\Longrightarrow} \langle P'', c'\rangle$ where $(x \sqcup b') = \alpha'$ and $c' \sqcup b' = e$. Using Lemma 6 we have that $(x, c') \in \mathcal{M}(\langle P, c\rangle)$, now we will prove that $(x, c') \succeq (\alpha, \beta)$,

namely $x \sqsubseteq \alpha$ and $\beta \sqsubseteq (\alpha \sqcup c')$. Recall that $x \sqsubseteq \alpha' \sqsubseteq \alpha$, now for the latter condition $(\alpha \sqcup c') = (\alpha' \sqcup b \sqcup c') = (x \sqcup b' \sqcup b \sqcup c') = (x \sqcup b \sqcup e)$ then since $d'' \sqsubseteq e$ we can check that $\beta \sqsubseteq d' \sqsubseteq (d' \sqcup x) = (d'' \sqcup b \sqcup x) \sqsubseteq (e \sqcup b \sqcup x) = (\alpha \sqcup c')$. Thus, this would mean that (α, β) is irrelevant in $\mathcal{M}(\langle P, c\rangle)$, a contradiction, therefore $\alpha' = \alpha$ and by consequence $d'' = d'$. Therefore, we know that $\langle Q, d\rangle \stackrel{\alpha}{\Longrightarrow} \langle Q', d'\rangle$, now let us assume by contradiction that $d' \neq \beta$ (i.e. $\beta \sqsubset d'$). By soundness and Lemma 3 we have that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$, this means that $(\alpha, d') \in [\![\langle Q, d\rangle]\!]$. By hypothesis then $(\alpha, d') \in [\![\langle P, c\rangle]\!]$ so there exist P_1, c_1 s.t. $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P_1, c_1\rangle$ and $d' \sqsubseteq c_1$. By Lemma 3 then $\langle P, c \sqcup \alpha\rangle \Longrightarrow \langle P_1, c_1\rangle$, now by completeness, there exist y, b'' s.t. $\langle P, c\rangle \stackrel{y}{\Longrightarrow} \langle P_1, c_1'\rangle$ where $y \sqcup b'' = \alpha$ and $c_1' \sqcup b'' = c_1$. Using Lemma 6 we get that $(y, c_1') \in \mathcal{M}(\langle P, c\rangle)$. Now let us prove that $(y, c_1') \succeq (\alpha, \beta)$, namely $y \sqsubseteq \alpha$ and $\beta \sqsubseteq (\alpha \sqcup c_1')$. The first condition follows from $y \sqcup b'' = \alpha$ and for the latter condition we proceed as follows $\beta \sqsubset d' \sqsubseteq c_1 \sqsubseteq (c_1 \sqcup y) = (c_1' \sqcup b'' \sqcup y) = (c_1' \sqcup \alpha)$. Again, this would mean that (α, β) is irrelevant in $\mathcal{M}(\langle P, c\rangle)$, a contradiction, therefore $d' = \beta$. Hence, we know that $\langle Q, d\rangle \stackrel{\alpha}{\Longrightarrow} \langle Q', \beta\rangle$ then by Proposition 6 $(\alpha, \beta) \in \mathcal{M}(\langle Q, d\rangle)$. Finally, let us assume by contradiction that (α, β) is irrelevant in $\mathcal{M}(\langle Q, d\rangle)$. Then there exists $(\alpha_1, \beta_1) \in \mathcal{M}(\langle Q, d\rangle)$ such that $(\alpha_1, \beta_1) \succeq (\alpha, \beta)$, namely $\alpha_1 \sqsubseteq \alpha$ (equivalently $\exists z'.\alpha_1 \sqcup z' = \alpha$) and $\beta \sqsubseteq \alpha \sqcup \beta_1$. By Proposition 7 we have that $\langle Q, d \sqcup \alpha_1\rangle \longrightarrow^* \langle Q_1, \beta_1\rangle$, then $(\alpha_1, \beta_1) \in [\![\langle Q, d\rangle]\!]$ and by hypothesis $(\alpha_1, \beta_1) \in [\![\langle P, c\rangle]\!]$. This means that $\langle P, c \sqcup \alpha_1\rangle \longrightarrow^* \langle P_2, c_2\rangle$ where $\beta_1 \sqsubseteq c_2$, now by Lemma 3 we get $\langle P, c \sqcup \alpha_1\rangle \Longrightarrow \langle P_2, c_2\rangle$. By completeness of $\Longrightarrow$ there exist a, b_1 s.t. $\langle P, c\rangle \stackrel{a}{\Longrightarrow} \langle P_2, c_2'\rangle$ where $(a \sqcup b_1) = \alpha_1$ and $(c_2' \sqcup b_1) = c_2$. Hence, by Proposition 6 we know that $(a, c_2') \in \mathcal{M}(\langle P, c\rangle)$. Now let us prove that $(a, c_2') \succeq (\alpha, \beta)$ namely $a \sqsubseteq \alpha$ and $\beta \sqsubseteq (\alpha \sqcup c_2')$. First $a \sqsubseteq \alpha_1 \sqsubseteq \alpha$, for the latter condition we proceed as follows $(\alpha \sqcup c_2') = (\alpha_1 \sqcup z' \sqcup c_2') = (a \sqcup b_1 \sqcup z' \sqcup c_2') = (c_2 \sqcup z')$ and since $\beta_1 \sqsubseteq c_2$ and $\alpha \sqsubseteq c_2$ then $\beta \sqsubseteq (\alpha \sqcup \beta_1) \sqsubseteq (c_2 \sqcup z') = (\alpha \sqcup c_2')$. Once again, this would mean that (α, β) is irrelevant in $\mathcal{M}(\langle P, c\rangle)$, a contradiction. Finally, we can conclude that (α, β) is relevant in $\mathcal{M}(\langle Q, d\rangle)$ therefore $(\alpha, \beta) \in \mathcal{M}^C(\langle Q, d\rangle)$. □

Using the these Lemma 6 and 7 we conclude the following theorem.

Theorem 6. $[\![\langle P, c\rangle]\!] = [\![\langle Q, d\rangle]\!]$ *iff* $\mathcal{M}^C(\langle P, c\rangle) = \mathcal{M}^C(\langle Q, d\rangle)$

By combining Theorem 5 and Theorem 6 we get a simple decision procedure for $\dot{\approx}_{sb}$ by reducing weak saturated equivalence between two given configuration to the set equivalence of the corresponding compact input-output representations. The complexity of this procedure is clearly determined by the complexity of constructions of the compact input-output sets.

Theorem 7. *Let γ and γ' be two CCP\+ configurations. Let $IS = \{\gamma, \gamma'\}$ and let $N = |\texttt{Config}_{\longrightarrow}(IS)|$. Then*

- $\mathcal{M}^C(\gamma) = \mathcal{M}^C(\gamma')$ *iff* $\gamma \mathrel{\dot{\approx}}_{sb} \gamma'$.
- *Checking whether $\mathcal{M}^C(\gamma) = \mathcal{M}^C(\gamma')$ takes $O(N^2)$ time and uses $O(N)$ space.*

Proof. The first item follows from Theorem 5 and Theorem 6 and the second item is derived from the construction of $\mathcal{M}^C(\gamma)$ and $\mathcal{M}^C(\gamma')$. □

6. Improving the partition refinement for CCP

In this section we show that in the general case of CCP systems, the strategy from Section 4.2 can be used for their CCP\+ components, thus producing a $IS^{\star}_{\rightsquigarrow}$ which may be significant smaller (although the worst case remains exponential).

Given a configuration γ the idea is to detect when an evolution of γ, i.e. a γ' s.t. $\gamma \stackrel{\alpha_1}{\Longrightarrow} \ldots \stackrel{\alpha_k}{\Longrightarrow} \gamma'$, is a CCP\+ configuration. This way we can avoid adding new configurations with Rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ whenever $\gamma' \in Conf_{\texttt{CCP}\backslash+}$, and redundancy can be then checked using Proposition 3.

$$(\mathsf{IS'} \Longrightarrow) \frac{\gamma \in IS}{\gamma \in IS^{\star}_{\rightsquigarrow}} \quad (\mathsf{RS'} \Longrightarrow) \frac{\gamma \in IS^{\star}_{\rightsquigarrow} \;\; \gamma \stackrel{\alpha}{\rightsquigarrow} \gamma'}{\gamma' \in IS^{\star}_{\rightsquigarrow}}$$

$$(\mathsf{opt\text{-}RD} \Longrightarrow) \frac{\begin{array}{c}\gamma \in IS^{\star}_{\rightsquigarrow} \quad \gamma \notin Conf_{\texttt{CCP}\backslash+} \quad t_1 = \gamma \stackrel{\alpha}{\rightsquigarrow} \langle P_1, c_1\rangle \\ t_2 = \gamma \stackrel{\beta}{\rightsquigarrow} \langle P_2, c_2\rangle \quad \alpha \sqsubset \beta \quad c_2 = c_1 \sqcup \beta\end{array}}{\langle P_1, c_2\rangle \in IS^{\star}_{\rightsquigarrow}}$$

Table 5: Rules for improved version of the partition refinement for CCP.

Definition 15. *(Improved partition refinement for CCP) We define the procedure* `imp-weak-pr-ccp`$(IS, \rightsquigarrow)$ *by using the rules in Table 5 in Step 1 of* `weak-pr-ccp`$(IS, \rightsquigarrow)$ *from Algorithm 3.*

Using this algorithm we can decide $\mathrel{\dot{\approx}}_{sb}$ in a more efficient manner, although, in the worst-case scenario, still with exponential time. This follows from Proposition 3 and Theorem 1.

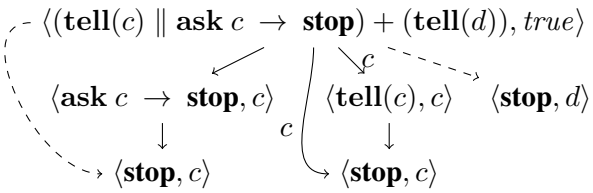

Figure 5: The above example illustrates the advantages of using the method discussed in Section 6: only the dashed transitions need to be taken into account to check $\dot{\approx}_{sb}$. Transitions are computed according to $\Longrightarrow$ (Table 3) and self-loops are omitted.

Theorem 8. *Let γ and γ' be two CCP configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of* `imp-weak-pr-ccp`$(IS, \Longrightarrow)$ *in Definition 15. Then*

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

It is clear that it is better to use `imp-weak-pr-ccp`$(IS, \Longrightarrow)$ instead of `weak-pr-ccp`$(IS, \Longrightarrow)$ since the new procedure avoids adding new states whenever they are not necessary to check redundancy w.r.t. $\dot{\approx}_{sb}$. In other words, if a configuration has subcomponents which do not use the nondeterministic choice then we can take advantage of this by efficiently deleting their redundant transitions.

Unfortunately, this improvement does not escape from the worst-case scenario of `weak-pr-ccp`$(IS, \Longrightarrow)$. Nevertheless, this approach shows the applicability of the strategy developed in Section 4.2. One may expect a practical impact of this optimization since most configurations are composed by several CCP\+ subconfigurations. We want to conclude this section by pointing to Figure 5 which illustrates the main advantage of using this method.

7. Congruence issues

A typical question in the realm of process calculi, and concurrency in general, is whether a given process equivalence is a *congruence*. In other words, whether the fact that P and Q are equivalent implies that they are still equivalent in any context. More precisely, an equivalence $\cong$ is a congruence if $P \cong Q$ implies $C[P] \cong C[Q]$ for every process context C. Intuitively, a context C is an expression with a hole $\bullet$ such that replacing $\bullet$ with a process yields a process term. The

expression $C[P]$ denotes the process that results from replacing in C, the hole $\bullet$ with P. For example $C = R \parallel \bullet$ then $C[P] = R \parallel P$.

The congruence issue is fundamental for algebraic as well as practical reasons; one may not be content with having $P \cong Q$ equivalent but $R \parallel P \ncong R \parallel Q$. Nevertheless, some of the representative equivalences in concurrency are not congruences. For example, in CCS [19], trace equivalence and strong bisimilarity are congruences but weak bisimilarity is not because it is not preserved by summation contexts. So given a notion of equivalence one may wonder in what contexts the equivalence is preserved. For instance, the problem with weak bisimilarity can be avoided by using a somewhat less liberal summation called guarded-summation (see [30]).

In this section we show that *weak (saturated barbed) bisimilarity* ($\dot{\approx}_{sb}$) is a congruence for the main CCP fragment here considered; CCP\+. We also show that $\dot{\approx}_{sb}$ is not a congruence for the full language of CCP. Moreover, unlike CCS, we shall see that restricting the syntax to guarded summation will not prevent the problem. In fact, our counterexample reveals that the problem is intrinsic to CCP. Despite the negative result for the full language, here we also show that weak bisimilarity is still a useful notion: its underlying co-inductive technique (bisimulation) as well as the verification algorithms we introduced in previous sections can be used to establish the standard observational equivalence in CCP. More precisely, we will show that $\dot{\approx}_{sb}$ implies observational equivalence in CCP, and hence it give us a semi-decision procedure for the standard equivalence of CCP.

7.1. Observational Equivalence

The standard notion of observables for CCP are the *results* computed by a process for a given initial store. More formally, given a (maximal) computation ξ of the form:

$$\langle Q_0, d_0\rangle \longrightarrow \langle Q_1, d_1\rangle \longrightarrow \ldots \longrightarrow \langle Q_n, d_n\rangle \not\longrightarrow$$

the result of ξ, denoted by $\mathsf{Result}(\xi)$, is the final store d_n.

We shall say then that two configurations are observational equivalent if they produce the same set of outputs when given the same input. More formally:

Definition 16. *(Observational equivalence) Let $\mathcal{O} : Proc \to Con_0 \to 2^{Con_0}$ be given by $\mathcal{O}(P)(d) = \{e \mid \langle P, d\rangle \longrightarrow^* \langle P', e\rangle \not\longrightarrow\}$. We say that P and Q are observational equivalent, written $P \sim_o Q$, iff for all d, $\mathcal{O}(P)(d) = \mathcal{O}(Q)(d)$.*

In [8, 9] it was shown that, in CCP\+, weak saturated barbed bisimilarity and observation equivalence coincide. Recall that $P \dot{\approx}_{sb} Q$ means $\langle P, true\rangle \dot{\approx}_{sb} \langle Q, true\rangle$.

Theorem 9. *([8, 9]) Let P and Q be CCP\+ processes. Then $P \sim_o Q$ if and only if $P \dot{\approx}_{sb} Q$.*

Nevertheless, the above theorem does not hold for the full language of CCP. We can show this by using a counter-example reminiscent from the standard one for CCS. Namely, in CCS one can prove that, in general, $a + \tau.P$ is not weakly bisimilar to $a + P$. We can use a similar approach for CCP as follows.

Claim 1. *There are P, Q s.t. $P \sim_o Q$ but $P \not\dot{\approx}_{sb} Q$.*

Proof. Let $P = (\mathbf{ask}\ (b)\ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (true)\ \rightarrow\ \mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e))$ and $Q = (\mathbf{ask}\ (b)\ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e))$ for unrelated elements $b, c, d, e \in Con_0$. It is straightforward to see that $P \sim_o Q$ since the only relevant inputs are $true, b$ and d for which both processes produce $true, c$ and e respectively. Now let us show that $P \not\dot{\approx}_{sb} Q$. First notice that if we take the transition $\langle P, true\rangle \longrightarrow \langle \mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e), true\rangle$ then $\langle Q, true\rangle$ can only stay still. Now if we take the transition $\langle Q, true\rangle \xrightarrow{b} \langle \mathbf{tell}(c), b\rangle$ then note that $\langle \mathbf{tell}(c), b\rangle \Downarrow_c$ but $\langle \mathbf{ask}\ (d)\ \rightarrow\ \mathbf{tell}(e), b\rangle \not\Downarrow_c$. Hence $P \not\dot{\approx}_{sb} Q$. □

However, the "if" direction of the theorem does hold. We show this next.

Theorem 10. *If $P \dot{\approx}_{sb} Q$ then $P \sim_o Q$.*

Proof. Let us assume by means of contradiction that $P \dot{\approx}_{sb} Q$ and there exists d s.t. $\mathcal{O}(P)(d) \neq \mathcal{O}(Q)(d)$. By definition, this means that there is an α s.t. $\alpha \in \mathcal{O}(P)(d)$ but $\alpha \notin \mathcal{O}(Q)(d)$. Hence $\langle P, d\rangle \longrightarrow^* \langle P', \alpha\rangle \not\longrightarrow$ however there is no Q' s.t. $\langle Q, d\rangle \longrightarrow^* \langle Q', \alpha\rangle \not\longrightarrow$. Therefore, for each computation $\langle Q, d\rangle \longrightarrow^* \langle Q', \beta\rangle \not\longrightarrow$ we have that either (i) $\beta \sqsubset \alpha$, (ii) $\alpha \sqsubset \beta$ or (iii) $\beta \not\sqsubseteq \alpha$. We shall prove that there is no saturated barbed bisimulation $\mathcal{R}$ containing the pair $\langle P', \alpha\rangle \mathcal{R} \langle Q', \beta\rangle$ for any of the three cases of β. First, consider the computations of type (i). Notice that since $\langle Q', \beta\rangle \not\longrightarrow$ and $\beta \sqsubset \alpha$ then $\langle Q', \beta\rangle \not\Downarrow_\alpha$ while $\langle P', \alpha\rangle \Downarrow_\alpha$. Now take the type (ii) and this time $\langle Q', \beta\rangle \Downarrow_\beta$ but $\langle P', \alpha\rangle \not\Downarrow_\beta$ since $\alpha \sqsubset \beta$. Similarly, in the type (iii) computations we have that $\langle Q', \beta\rangle \not\Downarrow_\alpha$ however $\langle P', \alpha\rangle \Downarrow_\alpha$. From these three cases we can conclude that $\langle P', \alpha\rangle$ cannot be related with $\langle Q', \beta\rangle$ in $\mathcal{R}$. Finally, in order to match $\langle P', \alpha\rangle \Downarrow_\alpha$ then the only case left would correspond to $\langle Q, d\rangle \longrightarrow^* \langle Q', \alpha\rangle \longrightarrow^* \langle Q'', \alpha'\rangle$ with $\alpha \sqsubset \alpha'$, but again $\langle Q', \alpha\rangle \Downarrow_{\alpha'}$ while $\langle P', \alpha\rangle \not\Downarrow_{\alpha'}$. Since $\langle Q, d\rangle$ is not able to match $\langle P, d\rangle \longrightarrow^*$

$\langle P', \alpha \rangle$ then there cannot be weak saturated barbed bisimulation relating $\langle P, d \rangle$ and $\langle Q, d \rangle$, a contradiction to the hypothesis $P \dot{\approx}_{sb} Q$. □

Therefore, we can use the algorithms presented in this paper as semi-decision procedures for observational equivalence. We can also use the co-inductive technique of weak saturated barbed bisimulation to establish observational equivalence: I.e., if we can exhibit a weak saturated barbed bisimulation containing the pair $\langle P, true \rangle$ and $\langle Q, true \rangle$ then $P \sim_o Q$.

7.2. Congruence

We begin by showing that weak bisimilarity is a congruence in a restricted sense: It is preserved by all the contexts from the summation-free fragment. For this purpose it is convenient to recall an equivalent definition of weak bisimilarity introduced in [8, 9].

Definition 17. *(Weak bisimilarity) A weak bisimulation is a symmetric relation $\mathcal{R}$ on configurations such that whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c \rangle$ and $\gamma_2 = \langle Q, d \rangle$:*

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

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ then $\exists \gamma_2'$ s.t. $\langle Q, d \sqcup \alpha \rangle \longrightarrow^* \gamma_2'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

We say that γ_1 and γ_2 are weakly bisimilar, written $\gamma_1 \dot{\approx} \gamma_2$, if there exists a weak bisimulation $\mathcal{R}$ such that $(\gamma_1, \gamma_2) \in \mathcal{R}$. We write $P \dot{\approx} Q$ iff $\langle P, true \rangle \dot{\approx} \langle Q, true \rangle$.

The following theorem from [8, 9] states that the above equivalence coincides with weak saturated barbed bisimilarity (Definition 4).

Theorem 11. *([8, 9]) $\dot{\approx}_{sb} = \dot{\approx}$.*

Using these definitions we can now prove that $\dot{\approx}_{sb}$ is a congruence in CCP\+.

Theorem 12. *Let P, Q and R be CCP\+ processes and assume that $P \dot{\approx}_{sb} Q$. Then*

1. $(\mathbf{ask}\ (c) \rightarrow P) \dot{\approx}_{sb} (\mathbf{ask}\ (c) \rightarrow Q)$,
2. $(P \parallel R) \dot{\approx}_{sb} (Q \parallel R)$.

Proof. We will focus on the parallel operator since the ask case is trivial. We shall prove that $\mathcal{R} = \{(\langle P \parallel R, c \rangle, \langle Q \parallel R, d \rangle) \mid \langle P, c \rangle \dot{\approx} \langle Q, d \rangle\}$ is a weak bisimulation (Definition 17). The result then follows from Theorem 11.

(i) Assume that $\langle P \parallel R, c\rangle \downarrow_\alpha$ then, since $\langle P, c\rangle \dot{\approx} \langle Q, d\rangle$, by condition (i) we have that $\langle Q \parallel R, d\rangle \Downarrow_\alpha$.

(ii) Now suppose that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c_1\rangle$ for some $\langle P_1, c_1\rangle$. By induction on (the depth) of the inference of $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c_1\rangle$ we have the following two cases:

- Using Rule LR3 (left) we have that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P' \parallel R, c'\rangle$, hence $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ by a shorter inference. Using this transition and the hypothesis in $\mathcal{R}$ we can conclude that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ and $\langle P', c'\rangle \dot{\approx} \langle Q', d'\rangle$ **(1)**. Now using Rule LR3 we get that $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d'\rangle$ and, because of **(1)**, we know that $\langle P' \parallel R, c'\rangle \mathcal{R} \langle Q' \parallel R, d'\rangle$.
- Using Rule LR3 (right) we have that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P \parallel R', c'\rangle$, hence $\langle R, c\rangle \xrightarrow{\alpha} \langle R', c'\rangle$ by a shorter inference, where $c' = c \sqcup \alpha \sqcup \beta$. Given that $\langle P, c\rangle \downarrow_c$ then by the hypothesis we know that $\langle Q, d\rangle \Downarrow_c$. This means that $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle \downarrow_c$, moreover by Lemma 4, Theorem 11 and the hypothesis we can conclude that $\langle Q, d\rangle \dot{\approx} \langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$. Therefore $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d' \sqcup \alpha\rangle$ since $c \sqsubseteq d'$. Now from this transition notice that $\langle Q' \parallel R, d' \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R', d' \sqcup \alpha \sqcup \beta\rangle$ and it is the case that $\langle Q' \parallel R', d' \sqcup \alpha \sqcup \beta\rangle \mathcal{R} \langle P \parallel R', c \sqcup \alpha \sqcup \beta\rangle$ since $\langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$.

□

Unfortunately, due to summation, the second point in the theorem above does not hold for arbitrary CCP processes.

Claim 2. *There are P', Q, R such that (a) $P' \dot{\approx}_{sb} Q$ and (b) $(P' \parallel R) \dot{\not\approx}_{sb} (Q \parallel R)$.*

To prove this claim we let $P = (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(c)) + (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(d))$, $P' = P \parallel \mathbf{tell}(e)$ and $Q = (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(c \sqcup e)) + (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(d \sqcup e))$ with $c \not\sqsubseteq d, c \not\sqsubseteq e, d \not\sqsubseteq c, d \not\sqsubseteq e, e \not\sqsubseteq c, e \not\sqsubseteq d$. For (a) we can show that $\langle P', true\rangle \dot{\approx}_{sb} \langle P, e\rangle \dot{\approx}_{sb} \langle Q, true\rangle$. The first equation is trivial. For the second we define a relation on configurations $\mathcal{R}$. The set of pairs in $\mathcal{R}$ are those linked in Figure 6. It can easily be verified that (the symmetric closure of) $\mathcal{R}$ is a weak bisimulation (see Definition 17). The point (a) then follows from Theorem 11.

For proving the part (b) of the above claim, we let $R = (\mathbf{ask}\ (e) \rightarrow \mathbf{tell}(\alpha)) + (\mathbf{ask}\ (e) \rightarrow \mathbf{tell}(\beta))$. We shall prove that no weak bisimulation can contain the

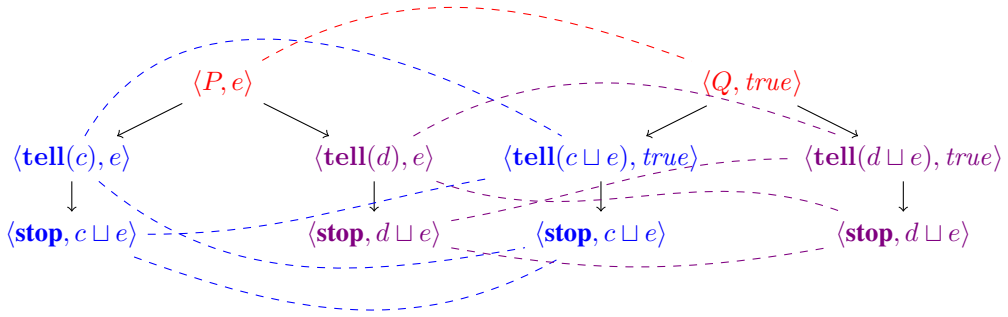

Figure 6: Let $P = (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(c)) + (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(d))$ and $Q = (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(c \sqcup e)) + (\mathbf{ask}\ (true) \rightarrow \mathbf{tell}(d \sqcup e))$. The linked configurations are weakly bisimilar.

pair $(\langle P \parallel R, e\rangle, \langle Q \parallel R, true\rangle)$. The results then follows from Theorem 11 and the fact that $\langle P' \parallel R, true\rangle \dot{\approx}_{sb} \langle P \parallel R, e\rangle$ which can be easily verified.

Consequently, let us assume that $\langle P \parallel R, e\rangle \longrightarrow \langle P \parallel \mathbf{tell}(\alpha), e\rangle$ by executing the left summand of R. By condition (ii) of weak bisimulation $\langle Q \parallel R, true\rangle$ must match the move. We have two cases:

- $\langle Q \parallel R, true\rangle$ does not make a transition. And now let us suppose that $\langle Q \parallel R, true\rangle \xrightarrow{e} \langle Q \parallel \mathbf{tell}(\beta), true\rangle$. This means that $\langle P \parallel \mathbf{tell}(\alpha), e\rangle$ now has to match this transition. However $\langle Q \parallel \mathbf{tell}(\beta), true\rangle \longrightarrow \langle Q, \beta\rangle \Downarrow_\beta$ while $\langle P \parallel \mathbf{tell}(\alpha), e\rangle \not\Downarrow_\beta$. Thus we cannot satisfy condition (i) of weak bisimulation.

- $\langle Q \parallel R, true\rangle$ makes a transition. To match the move it should also execute the left summand of R. However, since e is not the store of $\langle Q \parallel R, true\rangle$, Q must be executed first. and this means executing of one of summands in Q to be able to add e to the store. If the left summand of Q is executed, we get $\langle Q \parallel R, true\rangle \longrightarrow^* \langle \mathbf{tell}(\alpha), c \sqcup e\rangle$. In this case we could then take the move $\langle P \parallel \mathbf{tell}(\alpha), e\rangle \longrightarrow \langle \mathbf{tell}(d) \parallel \mathbf{tell}(\alpha), e\rangle$. But then $\langle \mathbf{tell}(\alpha), c \sqcup e\rangle \Downarrow_c$ and notice that $\langle \mathbf{tell}(d) \parallel \mathbf{tell}(\alpha), e\rangle \not\Downarrow_c$, thus we cannot satisfy condition (i) of weak bisimulation. The case where the right summand of Q is executed is symmetric.

□

8. Conclusions and Future Work

In this paper we explored the use of the partition refinement algorithm for CCP from [15] and [7] for checking observational equivalence in the CCP\+ fragment. In [15] authors gave a decision procedure for $\dot{\sim}_{sb}$ and in [7] authors proved how the algorithm for $\dot{\sim}_{sb}$ can be used to compute $\dot{\approx}_{sb}$. In this paper, we proved that this procedure takes exponential time and space (in the size of the set of reachable configurations) even for the restricted case of CCP\+. We then proposed two alternative methods for checking observational equivalence in CCP\+ by exploiting some of the intrinsic properties of this fragment, in particular confluence. We proved that both procedures take polynomial time (in the size of the set of reachable configurations), thus significantly improving the exponential-time approach from [15, 7], which is, to the best of our knowledge the only algorithm for checking observational equivalence in CCP. Each of the two method has its advantages over the other. On the one hand, the algorithm from Section 4 uses significantly more time and space than the one from Section 5, however it can be easily adapted for for full language of CCP as shown in Section 6. On the other hand, the procedure from Section 5 takes less time and uses only linear space nevertheless there is no "trivial" adaptation for the full language as it does not rely on the partition refinement approach. We also studied congruence issues for $\dot{\approx}_{sb}$ for CCP\+ and CCP as well as the relationship between $\dot{\approx}_{sb}$ and observational equivalence.

From a theoretical perspective, it would be interesting to lift such relationship to the whole CCP including recursive definitions: these allow infinite computations for which the notion of observation requires *fairness* [1]. In [8], we have shown that these notions coincides for CCP\+ but, for the whole calculus a better understanding is missing. From a more practical point of view, we would like to proceed with an experimental evaluation comparing the performances of the various approaches that we have discussed in this paper. Furthermore we would like to consider more efficient partition refinement algorithms [13] to check whether the algorithm from Section 4 can be further improved. The challenge would be to find a more efficient version of $\Longrightarrow$ that can still be used for deciding $\dot{\approx}_{sb}$ and so it can be adapted to the case of the full CCP. Finally, we plan to investigate how the procedures here defined can be extended to different versions of CCP where the summation operator is not present, for instance timed CCP (tcc) [31], universal temporal CCP (utcc) [32] and epistemic CCP (eccp) [6].

References

[1] 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.

[2] C. Palamidessi, V. A. Saraswat, F. D. Valencia, B. Victor, On the expressiveness of linearity vs persistence in the asychronous pi-calculus, in: 21th IEEE Symposium on Logic in Computer Science (LICS 2006), IEEE Computer Society, 2006, pp. 59–68. doi:10.1109/LICS.2006.39.

[3] M. G. Buscemi, U. Montanari, Open bisimulation for the concurrent constraint pi-calculus, in: S. Drossopoulou (Ed.), 17th European Symposium on Programming Languages and Systems (ESOP 2008), volume 4960 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 254–268. doi:10.1007/978-3-540-78739-6_20.

[4] J. Bengtson, M. Johansson, J. Parrow, B. Victor, Psi-calculi: Mobile processes, nominal data, and logic, in: 24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009), IEEE Computer Society, 2009, pp. 39–48. doi:10.1109/LICS.2009.20.

[5] M. Bartoletti, R. Zunino, A calculus of contracting processes, in: 25th Annual IEEE Symposium on Logic in Computer Science (LICS 2010), IEEE Computer Society, 2010, pp. 332–341. doi:10.1109/LICS.2010.25.

[6] 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_23.

[7] L. F. Pino, A. Aristizabal, F. Bonchi, F. Valencia, Weak CCP bisimilarity with strong procedures, Science of Computer Programming (2014). doi:10.1016/j.scico.2014.09.007.

[8] 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.

[9] A. Aristizábal, Bisimulation Techniques and Algorithms for Concurrent Constraint Programming, Ph.D. thesis, École Polytechnique, 2012.

[10] 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.

[11] J.-C. Fernandez, L. Mounier, Verifying bisimulations "On the Fly", in: J. Quemada, J. A. Mañas, E. Vázquez (Eds.), 3rd International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE 1990), North-Holland, 1990, pp. 95–110.

[12] A. Bouali, R. de Simone, Symbolic bisimulation minimisation, in: G. von Bochmann, D. K. Probst (Eds.), 4th International Workshop Computer Aided Verification (CAV 1992), volume 663 of *Lecture Notes in Computer Science*, Springer, 1992, pp. 96–108. doi:10.1007/3-540-56496-9_9.

[13] A. Dovier, C. Piazza, A. Policriti, An efficient algorithm for computing bisimulation equivalence, Theoretical Computer Science 311 (2004) 221–256. doi:10.1016/S0304-3975(03)00361-X.

[14] R. M. Amadio, I. Castellani, D. Sangiorgi, On bisimulations for the asynchronous pi-calculus., in: U. Montanari, V. Sassone (Eds.), 7th International Conference on Concurrency Theory (CONCUR 1996), volume 1119 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 147–162. doi:10.1007/3-540-61604-7_53.

[15] 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.

[16] L. Aceto, A. Ingolfsdottir, J. Srba, The algorithmics of bisimilarity, in: D. Sangiorgi, J. Rutten (Eds.), Advanced Topics in Bisimulation and Coin-

duction, Cambridge University Press, 2011, pp. 100–172. doi:10.1017/CBO9780511792588.

[17] H. Garavel, Reflections on the future of concurrency theory in general and process calculi in particular 209 (2008) 149–164. doi:10.1016/j.entcs.2008.04.009.

[18] L. F. Pino, F. Bonchi, F. D. Valencia, Efficient computation of program equivalence for confluent concurrent constraint programming, in: R. Peña, T. Schrijvers (Eds.), 15th International Symposium on Principles and Practice of Declarative Programming (PPDP 2013), ACM, 2013, pp. 263–274. doi:10.1145/2505879.2505902.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] F. Bonchi, U. Montanari, Symbolic semantics revisited, in: R. M. Amadio (Ed.), 11th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS 2008), volume 4962 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 395–412. doi:10.1007/978-3-540-78499-9_28.

[26] M. Hennessy, H. Lin, Symbolic bisimulations, Theorerical Computer Science 138 (1995) 353–389. doi:10.1016/0304-3975(94)00172-F.

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

[28] M. Pistore, D. Sangiorgi, A partition refinement algorithm for the π-calculus, Inf. Comput. 164 (2001) 264–321. doi:10.1006/inco.2000.2895.

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

[30] R. Milner, Communicating and mobile systems: the π-calculus, Cambridge University Press, 1999.

[31] V. A. Saraswat, R. Jagadeesan, V. Gupta, Foundations of timed concurrent constraint programming, in: 9th Annual Symposium on Logic in Computer Science (LICS 1994), IEEE Computer Society, 1994, pp. 71–80. doi:10.1109/LICS.1994.316085.

[32] C. Olarte, F. D. Valencia, Universal concurrent constraint programing: symbolic semantics and applications to security, in: R. L. Wainwright, H. Haddad (Eds.), 23rd Annual ACM Symposium on Applied Computing (SAC 2008), ACM, 2008, pp. 145–150. doi:10.1145/1363686.1363726.