Efficient Computation of Program Equivalence for Confluent Concurrent Constraint Programming[*]

Luis F. Pino
INRIA/DGA and LIX
École Polytechnique
91128 Palaiseau, France
luis.pino@lix.polytechnique.fr

Filippo Bonchi
CNRS and ENS Lyon
Université de Lyon, LIP
69364 Lyon, France
filippo.bonchi@ens-lyon.fr

Frank D. Valencia
CNRS and LIX
École Polytechnique
91128 Palaiseau, France
frank.valencia@lix.polytechnique.fr

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\+). We then significantly improve on the complexity of this algorithm by providing two alternative *polynomial-time* decision procedures for ccp\+ 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 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.

Categories and Subject Descriptors

D.3.2 [**Language Classifications**]: Constraint and logic languages. Concurrent, distributed, and parallel languages; D.2.4 [**Software / Program Verification**]: Formal methods; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Logic and constraint programming*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*Program analysis*

[*]This work has been partially supported by the project ANR 12IS02001 PACE, ANR-09-BLAN-0169-01 PANDA, and by the French Defence procurement agency (DGA) with a PhD grant.

PPDP '13, September 16 - 18 2013, Madrid, Spain
Copyright 2013 ACM 978-1-4503-2154-9/13/09 ...$15.00.
http://dx.doi.org/10.1145/2505879.2505902.

General Terms

Algorithms, Theory, Verification

Keywords

Concurrent Constraint Programming, Bisimulation, Partition Refinement, Observational Equivalence

1. INTRODUCTION

Motivation. *Concurrent constraint programming (ccp)* [26] 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 [23, 11, 7, 6, 17] 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 [17].

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 [5] 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 [26]. 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* [26], 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 [3], 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. In this paper we shall mainly deal with the verification of $\dot{\approx}_{sb}$ for arbitrary ccp\+ programs since, as mentioned above, $\dot{\approx}_{sb}$ coincides with observational program equivalence [3].

Unfortunately, the standard algorithms for checking bisimilarity (such as [16, 14, 10, 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 [2] where an input transition can be matched also by an internal (tau) transition.)

Partition Refinement for ccp. By building upon [2], we introduced in [4] 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 [4] 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 [4] 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 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 [5], this approach (which is referred in [1] as saturation) is proven to be unsound for ccp (see [5]). 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 ambient), 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 $\dot{\approx}_{sb}$, confluency 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 [16, 14, 10, 13]. Since redundancy can be determined statically, the additional states needed by the algorithm in [4] 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.

Contributions. The main contribution of this paper is the introduction of two novel decision procedures that can be used to decide the standard notion of program equivalence for ccp\+ in polynomial time. This represents a significant improvement over the previous algorithm for program equivalence, which, as we show in this paper, has an exponential time complexity even in the restricted case of ccp\+ programs. Each of these two new procedures has an advantage over the other. One has a better time complexity. The other can be easily adapted for the full language of ccp to produce significant state space reductions.

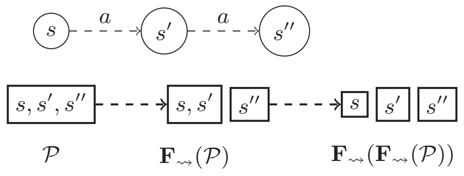
A technical report with detailed proofs of this paper can be found in [24]. We wish to conclude this introduction with a quote from [15] that captures the goal of the present paper:

"*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*".

Structure of the paper. The paper is organized as follows: In Section 2 we recall the basic knowledge concerning the standard partition refinement and the ccp formalism. In Section 3 we present the partition refinement for ccp from [4] and how it can be used to decide observational equivalence following [5]. Our contributions begin in Section 4 where we prove that the partition refinement for ccp from Section 3 is inefficient even for ccp\+, then we 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. Finally, in Section 7 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



s
a
s'
a
s''
s, s', s''
s, s'
s''
s
s'
s''
$\mathcal{P}$
$\mathbf{F}_{\rightsquigarrow}(\mathcal{P})$
$\mathbf{F}_{\rightsquigarrow}(\mathbf{F}_{\rightsquigarrow}(\mathcal{P}))$

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

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 $\{B_1\} \dots \{B_n\}$ to denote a partition consisting of (non-empty) blocks $B_1, \dots, B_n$. A partition represents an equivalence relation where equivalent elements belong to the same block. We write $s\mathcal{P}r$ to mean that s and r are equivalent in the partition $\mathcal{P}$.

LTSs and Graphs.

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

2.1 Partition Refinement

We report the partition refinement algorithm [16] 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

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

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]).

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 [12, 18] 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.

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 $\mathcal{P}^n$.
```

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 [3] 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\+). Henceforth, we use ccp\+ 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 $\gamma, \gamma', \dots$ to denote the set of all configurations. We will use $Conf_{\mathtt{ccp}\backslash+}$ for the ccp\+ configurations.

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 [3], the authors introduced a *barbed semantics* for ccp. Barbed equivalences have been introduced in [20] 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* [9, 8]. 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. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Weak saturated barbed bisimulations are defined as above by replacing $\downarrow$ by $\Downarrow$ and $\longrightarrow$ by $\longrightarrow^*$. 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. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Remark 3. It should be noticed that standard notion of observational ccp program equivalence was shown to coincide with $\dot{\approx}_{sb}$ in the case of ccp\+. For the sake of space we shall not introduce the standard notion– see [3] for further details.

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 $Q = \mathbf{ask}\ (x < 5) \rightarrow T$. Now, $\langle P, true\rangle \not\dot{\sim}_{sb} \langle Q, true\rangle$, since $\langle P, x < 7\rangle \longrightarrow$, 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 somehow 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 3.

2.5 Labeled Semantics

In [3] we have shown that $\dot{\approx}_{sb}$ is *fully abstract* with respect to the standard observational equivalence from [26]. Unfortunately, the quantification over all constraints in condition (iii) of Definition 3 makes hard checking $\dot{\sim}_{sb}$ and $\dot{\approx}_{sb}$, since one should check infinitely many constraints. In order to avoid this problem we have introduced in [3] 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., $\langle P, d \sqcup \alpha\rangle \longrightarrow \langle P', d'\rangle$. As a consequence, the transitions labeled with the constraint $true$ are in one to one correspondence with the reductions defined in the previous section. For this reason, hereafter we will sometimes write $\longrightarrow$ to mean $\xrightarrow{true}$. Before formally introducing the labeled semantics, we fix some notation.

Notation 1. We will use $\rightsquigarrow$ to denote a generic transition relation on the state space $Conf$ and labels Con_0. Also in this case $\rightsquigarrow$ mean $\overset{true}{\rightsquigarrow}$. Given a set of initial configurations IS, $\mathtt{Config}_{\rightsquigarrow}(IS)$ denote the set $\{\gamma' \mid \exists \gamma \in IS \text{ s.t. } \gamma \overset{\alpha_1}{\rightsquigarrow} \dots \overset{\alpha_n}{\rightsquigarrow} \gamma' \text{ for some } n \geq 0\}$.

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., $\alpha \in \min\{a \in Con_0 \mid c \sqsubseteq d \sqcup a\}$. 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.

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 $\langle P, true\rangle \overset{x<5}{\not\longrightarrow}$.

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

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* [4], which coincide with $\dot{\sim}_{sb}$ and, in the weak case [5], with $\dot{\approx}_{sb}$. This fact allowed us in [4] to define a variation of the partition refinement algorithm which we show in the next section.

First, we recall some results from [4] and [5], which are fundamental for the development of the paper.

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

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

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

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

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

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

Table 3: Weak semantics for ccp

Lemma 1. ([3], [5]) (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 [5].

Lemma 2. ([5]) (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 intutively 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.

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 [4] we adapted the standard partition refinement procedure to decide strong bisimilarity for ccp ($\dot{\sim}_{sb}$). As we did for the standard partition refinement, we 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 $\mathcal{P}^0 = \{B_1\} \dots \{B_m\}$, 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 4. (Transition Domination) Let t and t' be two transitions of the form $t = (\gamma, \alpha, \langle P', c'\rangle)$ and $t' = (\gamma, \beta, \langle P', c''\rangle)$. We say that t dominates t', written $t \succ_D t'$, iff $\alpha \sqsubset \beta$ 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. Let $P = (\mathbf{ask}\ (x < 15)\ \rightarrow\ \mathbf{tell}(y > 42)) + (\mathbf{ask}\ (x < 10)\ \rightarrow\ \mathbf{tell}(y > 42))$ and let $\gamma = \langle P, true\rangle$. Consider $t_1 = \gamma \xrightarrow{x<15} \langle \mathbf{tell}(y > 42), x < 15\rangle$ and $t_2 = \gamma \xrightarrow{x<10} \langle \mathbf{tell}(y > 42), x < 10\rangle$, then one can check that $t_1 \succ_D t_2$ since $(x < 15) \sqsubset (x < 10)$ and $(x < 10) = ((x < 15) \sqcup (x < 10))$.

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

To understand this definition better consider the following example.

Example 8. Let $Q_1 = (\mathbf{ask}\ (b)\ \rightarrow\ (\mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d)))$, $Q_2 = (\mathbf{ask}\ (a)\ \rightarrow\ \mathbf{stop})$ and $P = Q_1 + Q_2$, where $d \sqsubseteq c$ and $a \sqsubset b$. Now let $\gamma = \langle P, true\rangle$, then consider $t = \gamma \xrightarrow{a} \langle \mathbf{stop}, a\rangle$ and $t' = \gamma \xrightarrow{b} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), b\rangle$. 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 5. We have that $t \succ_D t''$ follows from $a \sqsubset b$. And we know $tar(t'')\mathcal{R}\ tar(t')$ from Example 4, i.e. $\langle \mathbf{stop}, b\rangle \dot{\approx}_{sb} \langle \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d), b\rangle$.

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

$T = \mathbf{tell}(\mathit{true})$ $\quad P = \mathbf{ask}\,(x < 7) \rightarrow T$ $\quad Q = \mathbf{ask}\,(x < 5) \rightarrow T$ $\quad R = \mathbf{ask}\,(z < 5) \rightarrow (P + Q)$

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

$$\begin{tikzcd}
\langle R'+S, \mathit{true}\rangle \arrow[r, "z<5"] \arrow[dr, "z<7"] & \langle P+Q', z<5\rangle \arrow[r, "x<5"] \arrow[ddd, bend right=60, "x<7"'] & \langle T', z<5 \sqcup x<5\rangle \arrow[r, "\mathit{true}"] & \langle \mathbf{stop}, z<5 \sqcup x<5 \sqcup y=1\rangle \\
\langle S, \mathit{true}\rangle \arrow[r, "z<7"] & \langle P, z<7\rangle \arrow[r, "x<7"] & \langle T, z<7 \sqcup x<7\rangle \arrow[r, "\mathit{true}"] & \langle \mathbf{stop}, z<7 \sqcup x<7\rangle \\
\langle R+S, \mathit{true}\rangle \arrow[r, "z<5"] \arrow[ur, "z<7"] & \langle P+Q, z<5\rangle \arrow[r, "x<5"] \arrow[dr, "x<7"] & \langle T, z<5 \sqcup x<5\rangle \arrow[r, "\mathit{true}"] & \langle \mathbf{stop}, z<5 \sqcup x<5\rangle \\
 & \langle P, z<5\rangle \arrow[r, "x<7"] & \langle T, z<5 \sqcup x<7\rangle \arrow[r, "\mathit{true}"] & \langle \mathbf{stop}, z<5 \sqcup x<7\rangle
\end{tikzcd}$$

Figure 2: $LTS_{\longrightarrow}(IS)$ **where (**$IS = \{\langle R' + S, \mathit{true}\rangle, \langle S, \mathit{true}\rangle, \langle R + S, \mathit{true}\rangle, \langle P, z < 5\rangle\}$**).**

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

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

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 5 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 [4] 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:

Definition 6. (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 6
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

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

Theorem 1. ([4]) 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 $\mathtt{Config}_{\longrightarrow}(IS)$.

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

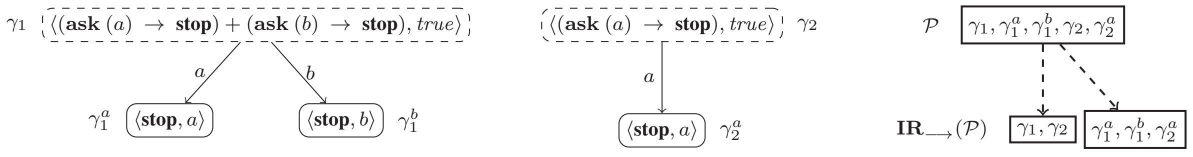
3.2 Weak equivalence

We can also use the above-mentioned algorithm to verify the weak version of saturated bisimilarity ($\dot{\approx}_{sb}$). Recall that in [3] it was shown that in ccp\+, $\dot{\approx}_{sb}$ coincides with the standard notion of ccp program (observational) equivalence.

Following [1] 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 Algorithm 2.

Definition 7. (Weak Partition Refinement for ccp) We define the procedure $\texttt{weak-pr-ccp}(IS, \rightsquigarrow)$ by replacing the barbs ($\downarrow_c$) in step 2 of Algorithm 2 with weak barbs ($\Downarrow_c$).

Using this algorithm we can decide $\dot{\approx}_{sb}$ also with exponential time. This follows from Theorem 1.



γ_1 $\langle(\mathbf{ask}\ (a) \rightarrow \mathbf{stop}) + (\mathbf{ask}\ (b) \rightarrow \mathbf{stop}), true\rangle$
a
b
γ_1^a $\langle\mathbf{stop}, a\rangle$
$\langle\mathbf{stop}, b\rangle$ γ_1^b
$\langle(\mathbf{ask}\ (a) \rightarrow \mathbf{stop}), true\rangle$ γ_2
a
$\langle\mathbf{stop}, a\rangle$ γ_2^a
$\mathcal{P}$ $\gamma_1, \gamma_1^a, \gamma_1^b, \gamma_2, \gamma_2^a$
$\mathbf{IR}_{\longrightarrow}(\mathcal{P})$ γ_1, γ_2 $\gamma_1^a, \gamma_1^b, \gamma_2^a$

Figure 3: An example of the use of $\mathbf{IR}_{\longrightarrow}(\mathcal{P})$ as in Definition 6. 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.

Theorem 2. ([5]) Let γ and γ' be two ccp configurations. Let $IS = \{\gamma, \gamma'\}$ and let $\mathcal{P}$ be the output of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$ in Definition 7. Then

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

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 $P^n = P_0^n$ with P_i^n, for $i \in \{0, \dots, n-1\}$, given by:

$P_i^n = (\mathbf{ask}\ (a_i) \rightarrow (\mathbf{ask}\ (b_i) \rightarrow P_{i+1}^n)) \parallel (\mathbf{ask}\ (b_i) \rightarrow \mathbf{stop})$

and $P_n^n = \mathbf{tell}(b_n)$. Furthermore, we assume that for all $i \in \{0, \dots, n-1\}$ we have $a_i \sqsubseteq b_i$ and for all $j \in \{0, \dots, n-1\}$ 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 $\gamma = \langle P^n, true\rangle$ and $IS = \{\gamma\}$, let $\mathcal{P}$ be the output $\texttt{pr-ccp}(IS, \longrightarrow)$ in 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.

Proposition 2. ([26]) Let $\gamma \in Conf_{\texttt{ccp}\backslash+}$. 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 configuration is weakly bisimilar to all its derivatives.

Lemma 3. Let $\gamma \in Conf_{\texttt{ccp}\backslash+}$. For all $\gamma' \in \mathsf{Deriv}(\gamma)$ we have $\gamma \dot{\approx}_{sb} \gamma'$.

PROOF. Let $\mathcal{R} = \{(\gamma_1, \gamma_2) \mid \exists\gamma_3 \text{ s.t. } \gamma_1 \longrightarrow^* \gamma_3 \text{ and } \gamma_2 \longrightarrow^* \gamma_3\}$, 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 $(\langle P_1, c_1 \sqcup e\rangle, \langle P_2, c_2 \sqcup e\rangle) \in \mathcal{R}$. $\square$

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 characteristics of the configurations of this fragment. In this section, using such features, we shall show that the complexity of $\texttt{weak-pr-ccp}(IS, \Longrightarrow)$ 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 5) becomes an easier task. As we explained in Section 3.1, the purpose of rule $(\mathsf{RD}^{IS}_{\leadsto})$ from Table 4 is to add some configurations to $IS^{\star}_{\leadsto}$ 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.

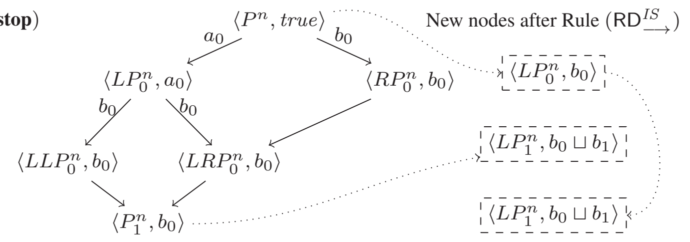
$$P^n = (\mathbf{ask}\,(a_0) \rightarrow (\mathbf{ask}\,(b_0) \rightarrow P^n_1)) \parallel (\mathbf{ask}\,(b_0) \rightarrow \mathbf{stop})$$

$$LP^n_0 = (\mathbf{ask}\,(b_0) \rightarrow P^n_1) \parallel (\mathbf{ask}\,(b_0) \rightarrow \mathbf{stop})$$

$$RP^n_0 = (\mathbf{ask}\,(a_i) \rightarrow (\mathbf{ask}\,(b_i) \rightarrow P^n_{i+1})) \parallel \mathbf{stop})$$

$$LLP^n_0 = P^n_1 \parallel (\mathbf{ask}\,(b_0) \rightarrow \mathbf{stop})$$

$$LRP^n_0 = (\mathbf{ask}\,(b_0) \rightarrow P^n_1) \parallel \mathbf{stop}$$



$\langle P^n, true\rangle$
a_0
b_0
New nodes after Rule $(\mathsf{RD}^{IS}_{\longrightarrow})$
$\langle LP^n_0, a_0\rangle$
$\langle RP^n_0, b_0\rangle$
$\langle LP^n_0, b_0\rangle$
b_0
b_0
$\langle LLP^n_0, b_0\rangle$
$\langle LRP^n_0, b_0\rangle$
$\langle LP^n_1, b_0 \sqcup b_1\rangle$
$\langle P^n_1, b_0\rangle$
$\langle LP^n_1, b_0 \sqcup b_1\rangle$

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.**

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

The definition above reflects the fact that when $\gamma \overset{\alpha}{\Longrightarrow}_{\max} \gamma'$ 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 4. (Soundness) If $\langle P, c\rangle \overset{\alpha}{\Longrightarrow}_{\max} \langle P', c'\rangle$ 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 s.t. $\langle P, c\rangle \overset{\alpha}{\Longrightarrow}_{\max} \langle P', c''\rangle$ 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. $\square$

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. ([5]) Let γ and γ' be two ccp configurations. Let $IS = \{\gamma, \gamma'\}$, let $\mathcal{P}$ be the output $\texttt{weak-pr-ccp}(IS, \Longrightarrow_{\max})$ in Definition 7. Then

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

PROOF. Follows from the correctness of $\Longrightarrow_{\max}$ (Lemma 4), the results in [5] and Theorem 2. $\square$

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. ($\Rightarrow$) 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 3 $\langle P, c \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_1, c_1 \sqcup \beta\rangle$. 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 $t_1 \succ_D t'$ and $\langle P_1, c_1 \sqcup \beta\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$.

($\Leftarrow$) Assume that $t_1 \succ_{\dot{\approx}_{sb}} t_2$ then there exists $t' = (\gamma, \beta, \langle P_1, c'\rangle)$ such that $t_1 \succ_D t'$ and $\langle P_1, c'\rangle \dot{\approx}_{sb} \langle P_2, c_2\rangle$. By $t_1 \succ_D t'$ 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 $\langle P_1, c'\rangle \longrightarrow^* \langle P', c_2\rangle \not\longrightarrow$. $\square$

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 3, Step 2).

Algorithm 3 $\texttt{weak-pr-dccp}(IS)$

```
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 $\gamma$ and $\gamma'$
     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 3 can be used to decide $\dot{\approx}_{sb}$ for configurations in $Conf_{\texttt{ccp}\backslash+}$ 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 3 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. [22], this step takes $O(|E|\log|V|)$ time and uses $O(|E|)$ space. Therefore, since $|\mathtt{V}(G)| = N$ and $|\mathtt{E}(G)| = N^2$, 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 3, 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 3.

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 3 which itself depends on the confluent nature of ccp\+.

Theorem 5. $\langle P, c\rangle \dot{\approx}_{sb} \langle Q, d\rangle$ 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 3 $\langle P, c\rangle \dot{\approx}_{sb} \langle P', c'\rangle$ 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 $\langle P, c\rangle \sim_{wb} \langle Q, d\rangle$

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 $(true, true) \succeq (\alpha, \alpha)$ but $(true, true) \not\succeq (\alpha \sqcup \beta, \alpha \sqcup \beta \sqcup c)$. 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 $(\beta, e') \in [\![\langle P, c\rangle]\!]$.

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

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

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:

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

where $\otimes : \mathit{Con}_0 \times 2^{\mathit{Con}_0 \times \mathit{Con}_0} \longrightarrow 2^{\mathit{Con}_0 \times \mathit{Con}_0}$ 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 $\gamma = \langle \mathbf{ask}\,(\alpha) \rightarrow (\mathbf{ask}\,(\beta) \rightarrow \mathbf{tell}(c)), \mathit{true}\rangle$ and $\gamma' = \langle \mathbf{ask}\,(\alpha \sqcup \beta) \rightarrow \mathbf{tell}(c), \mathit{true}\rangle$ where $\alpha, \beta, c \in \mathit{Con}_0$ 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:

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

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

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:

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

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:

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

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 $\langle P, c\rangle \xRightarrow{\alpha} \langle P', c'\rangle$ then $(\alpha, c') \in \mathcal{M}(\langle P, c\rangle)$.

The following proposition states that whenever a pair (α, e) belongs to $\mathcal{M}(\langle P, c\rangle)$, 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 $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', e\rangle$

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 5. 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. (Sketch) Assume that $(\alpha, \beta) \in [\![\langle P, c\rangle]\!]$ then by definition $\langle P, c \sqcup \alpha\rangle \longrightarrow^* \langle P', \beta'\rangle$ s.t. $\beta \sqsubseteq \beta'$. Using soundness and completeness of $\Longrightarrow$ (Lemma 2) there exists $(\alpha', c') \in \mathcal{M}(\langle P, c\rangle)$ where $\alpha' \sqcup b = \alpha$ and $c' \sqcup b = \beta'$. By well-foundedness of $\succeq$ (Proposition 5) there is a relevant $(\alpha'', c'') \in \mathcal{M}^C(\langle P, c\rangle)$ s.t. $(\alpha'' \sqcup x) = \alpha'$ and $c' \sqsubseteq (c'' \sqcup \alpha')$. Since $(\alpha'', c'') \in \mathcal{M}^C(\langle P, c\rangle)$ then $(\alpha'', c'') \in \mathcal{M}^C(\langle Q, d\rangle)$ and so $(\alpha'', c'') \in \mathcal{M}(\langle Q, d\rangle)$. Therefore $\langle Q, d \sqcup \alpha''\rangle \longrightarrow^* \langle Q', c''\rangle$ and by monotonicity $\langle Q, d \sqcup \alpha'' \sqcup x \sqcup b\rangle \longrightarrow^* \langle Q', c'' \sqcup x \sqcup b\rangle$ which is equivalent to say that $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', c'' \sqcup x \sqcup b\rangle$. Finally, since $\beta \sqsubseteq (c'' \sqcup x \sqcup b)$ then $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$. The full proof can be found in the technical report [24]. $\square$

Lemma 6. 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. (Sketch) Assume that $(\alpha, \beta) \in \mathcal{M}^C(\langle P, c\rangle)$ and (α, β) is relevant in $\mathcal{M}(\langle P, c\rangle)$ **(1)**. Then $(\alpha, \beta) \in \mathcal{M}(\langle P, c\rangle)$ and using Proposition 7 we can conclude that $(\alpha, \beta) \in [\![\langle P, c\rangle]\!]$, hence by hypothesis $(\alpha, \beta) \in [\![\langle Q, d\rangle]\!]$. This means that there exists d' s.t. $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\beta \sqsubseteq d'$. By completeness of $\Longrightarrow$ (Lemma 2) there exists α' s.t. $\langle Q, d\rangle \xRightarrow{\alpha'} \langle Q', d''\rangle$. By contradiction, we assume that $\alpha' = \alpha$ and this we arrive to conclude that (α, β) is irrelevant in $\mathcal{M}(\langle P, c\rangle)$ (a contradiction with **(1)**). Hence $d'' = d'$, thus we have $\langle Q, d\rangle \xRightarrow{\alpha} \langle Q', d'\rangle$. Using the same reasoning by contradiction we can prove that $d' = \beta$, this way we have $\langle Q, d\rangle \xRightarrow{\alpha} \langle Q', \beta\rangle$. This means, by Proposition 6, that $(\alpha, \beta) \in \mathcal{M}(\langle Q, d\rangle)$. Finally, we can prove again by contradiction that (α, β) is relevant in $\mathcal{M}(\langle Q, d\rangle)$ and so $(\alpha, \beta) \in \mathcal{M}^C(\langle Q, d\rangle)$. The full proof can be found in the technical report [24]. $\square$

Using the these lemmata above 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 $N = |\mathtt{Config}_{\longrightarrow}(IS)|$. Then

- $\mathcal{M}^C(\gamma) = \mathcal{M}^C(\gamma')$ iff $\gamma \dot{\approx}_{sb} \gamma'$.
- 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 Follows from Theorem 5 and Theorem 6 and the second item is derived from the construction of $\mathcal{M}^C(\gamma)$ and $\mathcal{M}^C(\gamma')$. $\square$

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

$$(\mathsf{opt\text{-}RD}\Longrightarrow)\ \frac{\gamma \in IS^{\star}_{\rightsquigarrow} \quad \gamma \notin Conf_{\mathtt{ccp}\backslash+} \quad t_1 = \gamma \overset{\alpha}{\rightsquigarrow} \langle P_1, c_1\rangle \quad t_2 = \gamma \overset{\beta}{\rightsquigarrow} \langle P_2, c_2\rangle \quad \alpha \sqsubset \beta \quad c_2 = c_1 \sqcup \beta}{\langle P_1, c_2\rangle \in IS^{\star}_{\rightsquigarrow}}$$

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

$$\begin{tikzcd}
 & \langle(\mathbf{tell}(c) \parallel \mathbf{ask}\ c \rightarrow \mathbf{stop}) + (\mathbf{tell}(d)), true\rangle \arrow[dl] \arrow[d, "c"] \arrow[ddl, dashed, bend right=60] \arrow[dd, bend right, "c"'] \arrow[dr, dashed] & \\
\langle \mathbf{ask}\ c \rightarrow \mathbf{stop}, c\rangle \arrow[d] & \langle \mathbf{tell}(c), c\rangle \arrow[d] & \langle \mathbf{stop}, d\rangle \\
\langle \mathbf{stop}, c\rangle & \langle \mathbf{stop}, c\rangle &
\end{tikzcd}$$

Figure 5: This example above reflects the advantages of using the method discussed in Section 6 since only the dashed transitions need to be considered for checking $\dot{\approx}_{sb}$. Transitions are computed according to $\Longrightarrow$ (Table 3) and loops are omitted.

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. $\gamma \overset{\alpha_1}{\Longrightarrow} \ldots \overset{\alpha_k}{\Longrightarrow} \gamma'$, is a ccp\+ configuration. This way we can avoid adding new configurations with Rule $(\mathsf{RD}^{IS}_{\rightsquigarrow})$ whenever $\gamma' \in Conf_{\mathtt{ccp}\backslash+}$, and redundancy can be then checked using Proposition 3.

Definition 15. (Improved partition refinement for ccp) We define the procedure `imp-weak-pr-ccp`$(IS, \rightsquigarrow)$ by replacing the rules in Step 1 of `weak-pr-ccp`$(IS, \rightsquigarrow)$ from Definition 7 with the rules defined in Table 5.

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

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 $\mathtt{Config}_{\longrightarrow}(IS)$.

It is clear that `imp-weak-pr-ccp`$(IS, \Longrightarrow)$ performs better than `weak-pr-ccp`$(IS, \Longrightarrow)$ since the new procedure avoids adding new states whenever they are not necessary to check redundancy w.r.t. $\dot{\approx}_{sb}$. 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. Notice that this scenario arises with the use of the summation operator $(+)$ hence one may expect a practical impact on this improvement since most configurations are composed by several ccp\+ subconfigurations. We want to conclude this section with an example (Figure 5) that shows the main advantage of using this method.

7. CONCLUSIONS AND FUTURE WORK

In this paper we explored the use of the partition refinement algorithm for ccp from [4] and [5] for checking observational equivalence in the ccp\+ fragment. In [4] authors gave a decision procedure for $\dot{\sim}_{sb}$ and in [5] 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 [4, 5], 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 verifying observational equivalence for the full 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 since it does not use the partition refinement approach.

Most of the related work was already discussed in the introduction. As we mentioned in Section 4, it remains as a future work to consider more efficient partition refinement algorithms [13] to see 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) [25], universal temporal ccp (utcc) [21] and epistemic ccp (eccp) [17].

8. REFERENCES

[1] L. Aceto, A. Ingolfsdottir, and J. Srba. *Advanced Topics in Bisimulation and Coinduction*, chapter The Algorithmics of Bisimilarity, pages 100–172. Cambridge University Press, 2011.

[2] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous pi-calculus. In *CONCUR*, volume 1119 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 1996.

[3] A. Aristizabal, F. Bonchi, C. Palamidessi, L. Pino, and F. D. Valencia. Deriving labels and bisimilarity for concurrent constraint programming. In *FOSSACS*, LNCS, pages 138–152. Springer, 2011.

[4] A. Aristizabal, F. Bonchi, L. Pino, and F. D. Valencia. Partition refinement for bisimilarity in ccp. In *SAC*, pages 88–93. ACM, 2012.

[5] A. Aristizábal, F. Bonchi, L. F. Pino, and F. Valencia. Reducing weak to strong bisimilarity in ccp. In *ICE*, pages 2–16, 2012.

[6] M. Bartoletti and R. Zunino. A calculus of contracting processes. In *LICS*, pages 332–341. IEEE Computer Society, 2010.

[7] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *LICS*, pages 39–48, 2009.

[8] F. Bonchi, F. Gadducci, and G. V. Monreale. Reactive systems, barbed semantics, and the mobile ambients. In *FOSSACS*, LNCS, pages 272–287, 2009.

[9] F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *LICS*, pages 69–80. IEEE, 2006.

[10] A. Bouali and R. de Simone. Symbolic bisimulation minimisation. In *CAV*, volume 663 of *Lecture Notes in Computer Science*, pages 96–108. Springer, 1992.

[11] M. G. Buscemi and U. Montanari. Open bisimulation for the concurrent constraint pi-calculus. In *ESOP*, pages 254–268, 2008.

[12] F. S. de Boer, A. D. Pierro, and C. Palamidessi. Nondeterminism and infinite computations in constraint programming. *Theor. Comput. Sci.*, 151(1):37–78, 1995.

[13] A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theor. Comput. Sci.*, 311(1-3):221–256, 2004.

[14] J.-C. Fernandez and L. Mounier. Verifying bisimulations "on the fly". In *FORTE*, pages 95–110. North-Holland, 1990.

[15] H. Garavel. Reflections on the future of concurrency theory in general and process calculi in particular. In *Proc. of LIX Colloquium on Emergent Trends in Concurrency Theory*, Electr. Notes Theor. Comput. Sci. 209, pages 149–164, 2008.

[16] P. C. Kanellakis and S. A. Smolka. Ccs expressions, finite state processes, and three problems of equivalence. In *PODC*, pages 228–240. ACM, 1983.

[17] S. Knight, C. Palamidessi, P. Panangaden, and F. D. Valencia. Spatial and epistemic modalities in constraint-based process calculi. In *CONCUR*, pages 317–332, 2012.

[18] N. P. Mendler, P. Panangaden, P. J. Scott, and R. A. G. Seely. A logical view of concurrent constraint programming. *Nord. J. Comput.*, 2(2):181–220, 1995.

[19] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag New York, Inc., 1980.

[20] R. Milner and D. Sangiorgi. Barbed bisimulation. In *ICALP*, LNCS, pages 685–695. Springer, 1992.

[21] C. Olarte and F. D. Valencia. Universal concurrent constraint programing: symbolic semantics and applications to security. In *SAC*, pages 145–150. ACM, 2008.

[22] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, Dec. 1987.

[23] C. Palamidessi, V. A. Saraswat, F. D. Valencia, and B. Victor. On the expressiveness of linearity vs persistence in the asychronous pi-calculus. In *LICS*, pages 59–68, 2006.

[24] L. Pino, F. Bonchi, and F. Valencia. Efficient computation of program equivalence for confluent concurrent constraint programming (technical report). Technical report, LIX, Ecole Polytechnique, 2013. `http://www.lix.polytechnique.fr/~luis.pino/files/minset-extended.pdf`.

[25] V. A. Saraswat, R. Jagadeesan, and V. Gupta. Foundations of timed concurrent constraint programming. In *LICS*, pages 71–80. IEEE, 1994.

[26] V. A. Saraswat, M. C. Rinard, and P. Panangaden. Semantic foundations of concurrent constraint programming. In *POPL*, pages 333–352. ACM Press, 1991.