A Behavioral Congruence for Concurrent Constraint Programming with Non-deterministic Choice

Luis Fernando Pino Duque, Filippo Bonchi, Frank D. Valencia

A Behavioral Congruence for Concurrent Constraint Programming with Nondeterministic Choice*

Luis F. Pino[1], Filippo Bonchi[2], Frank D. Valencia[1]

[1] Comète, LIX, Laboratoire de l'École Polytechnique associé à l'INRIA
[2] CNRS - Laboratoire de l'Informatique du Parallélisme, ENS Lyon

Abstract. Concurrent constraint programming (ccp) is a well-established model of concurrency for reasoning about systems of multiple agents that interact with each other by posting and querying partial information on a shared space. (Weak) bisimilarity is one of the most representative notions of behavioral equivalence for models of concurrency. A notion of weak bisimilarity, called weak saturated bisimilarity ($\dot{\approx}_{sb}$), was recently proposed for ccp. This equivalence improves on previous bisimilarity notions for ccp that were too discriminating and it is a congruence for the choice-free fragment of ccp. In this paper, however, we show that $\dot{\approx}_{sb}$ is not a congruence for ccp with nondeterministic choice. We then introduce a new notion of bisimilarity, called weak full bisimilarity ($\approx_f$), and show that it is a congruence for the full language of ccp. We also show the adequacy of $\approx_f$ by establishing that it coincides with the congruence induced by closing $\dot{\approx}_{sb}$ under all contexts. The advantage of the new definition is that, unlike the congruence induced by $\dot{\approx}_{sb}$, it does not require quantifying over infinitely many contexts.

1 Introduction

The Context. Concurrency theory studies the description and the analysis of systems made of interacting *processes*. Processes are typically viewed as infinite objects, in the sense that they can produce arbitrary and possibly endless interactions with their environment. *Process calculi* treat these processes much like the λ-calculus treats computable functions. They provide a formal language in which processes are represented by terms, and a set of rewriting rules to represent process evolution (or transitions). For example, the term $P \parallel Q$ represents the process that results from the parallel composition of the processes P and Q. A (labeled) transition $P \xrightarrow{\alpha} P'$ represents the evolution of P into P' given an interaction α with the environment.

Concurrent Constraint Programming (ccp) [25,26] is a well-established formalism that combines the traditional algebraic and operational view of process calculi with a declarative one based upon first-order logic. Ccp processes can then be seen as computing agents as well as first-order logic formulae. In ccp, processes interact asynchronously by *posting* (or *telling*) and querying (or *asking*) information, traditionally referred to as *constraints*, in a shared-medium referred to as *the store*. Furthermore,

* 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.

ccp is parametric in a *constraint system* indicating interdependencies (entailment) between constraints and providing for the specification of data types and other rich structures. The above features have recently attracted a renewed attention as witnessed by the works [21,9,5,4] on calculi exhibiting data-types, logic assertions as well as tell and ask operations. More recently in [14] the authors proposed the post and ask interaction model of ccp as an abstraction of *social networks*.

In any computational model of processes, a central notion is that of *behavioral equivalences* [11]. These equivalences determine what processes are deemed indistinguishable and they are expected to be *congruences*. The congruence issue is of great importance for algebraic and *compositional* reasoning: If two processes are equivalent, one should be able to replace one with the other in any context and preserve the equivalence (see e.g, [13]). For example, if $\bowtie$ is a behavioral congruence, then $P \bowtie Q$ should imply $P \parallel R \bowtie Q \parallel R$.

Reasoning on processes and their equalities therefore means dealing with, and comparing, infinite structures. For this, a widely used mathematical tool is *coinduction* (see e.g. [1]). Coinduction is the dual of induction; while induction is a pervasive tool to reason about finite and stratified structures, coinduction offers similar strengths on structures that are circular or infinite. The most widely applied coinductive concept is *bisimulation*: *bisimilarity* is used to study behavioral equivalences, and the bisimulation proof method is used to prove such equivalences. In fact, most process calculi are equipped with a notion of bisimilarity.

The Problem. There have been few attempts to define notions of bisimilarity equivalence for ccp processes. These equivalences are, however, not completely satisfactory: As shown in [2], the one in [25] is too fine grained; i.e. it may tell apart processes whose logic interpretation is identical. The one in [16] quantifies over all possible inputs from the environment, and hence it is not clear whether it can lead to a feasible proof technique. The notion introduced in [2], called (weak) saturated barbed bisimilarity ($\dot{\approx}_{sb}$), solves the above-mentioned issues and it is a congruence for ccp without nondeterministic choice. Unfortunately, as we will show in this paper, it is not a congruence for the full language of ccp. In particular, in ccp with nondeterministic choice, $P \dot{\approx}_{sb} Q$ does not imply $P \parallel R \dot{\approx}_{sb} Q \parallel R$.

The goal of this paper is therefore to provide ccp with an adequate behavioral congruence based on the bisimulation proof method.

Our Approach. We build on a result of [2] showing that $\dot{\approx}_{sb}$ can be characterized by a novel bisimulation game (called, for simplicity, weak bisimulation) which relies at the same time on both *barbs* and *labeled transitions*. Barbs are basically predicates on the states, processes or configuration stating the observation we can make of them. This is rather peculiar with respect to the existing notions of bisimulations introduced for other process calculi where one usually exploits labeled transitions to avoid thinking about barbs and contexts. Indeed, labeled transitions usually capture barbs, in the sense that a state exposes a certain barb if and only if it performs a transition with a certain label. This is not the case of ccp, where barbs are observations on the store, while labeled transitions are determined by the processes. A more abstract understanding of

this peculiarity of ccp can be given within the framework of [7] which is an extension of [15] featuring barbs and weak semantics.

As it is customary for weak barbed equivalences, in our weak bisimulation game whenever a player exposes a barb $\downarrow_e$, the opponent should expose the weak barb $\Downarrow_e$, i.e. it should be able to reach a state satisfying $\downarrow_e$, but then the game restarts from the original state ignoring the arriving state. One of our contributions is to show that for ccp the arriving state cannot be ignored.

Our Contributions. In this work, we prove that $\dot{\approx}_{sb}$ is a congruence for ccp without nondeterministic choice but not for the full language of ccp. We then propose a new notion of bisimilarity, called *(weak) full bisimilarity* ($\approx_f$). We show that $\approx_f$ is a congruence for the full language of ccp. We also show the adequacy of the new notion by establishing that it is the largest congruence included in $\dot{\approx}_{sb}$. In other words $\approx_f$ coincides with the congruence induced by closing $\dot{\approx}_{sb}$ under all contexts. Beyond being a congruence, the advantage of $\approx_f$ is that it does not require quantifying over infinitely many contexts. This is also important as it may simplify decision procedures for the equivalence. To the best of our knowledge, this is the first behavioral equivalence, which does not appeal to quantification over arbitrary process contexts in its definition, that is a congruence for ccp with nondeterministic choice.

A technical report with detailed proofs of this paper can be found in [22].

Structure of the paper. The paper is organized as follows: In Section 2 we recall the ccp formalism. In Section 3 we introduce the standard notion of observational equivalence ($\sim_o$) for ccp (from [26]), we then show its relation with the weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$) (from [2]) for ccp with nondeterministic choice. We also prove that $\dot{\approx}_{sb}$ is not a congruence for the full ccp. In Section 4 we introduce our new notion $\approx_f$, and we prove that (i) $\approx_f$ coincides with $\dot{\approx}_{sb}$ in the choice-free fragment of ccp; (ii) $\approx_f$ is a congruence for ccp with summation; and (iii) $\approx_f$ coincides with the equivalence obtained after closing $\dot{\approx}_{sb}$ under any context. In Section 5 we present our conclusions and future work.

2 Background

We begin this section by recalling the notion of constraint system. We then present the concurrent constraint programming (ccp) formalism.

2.1 Constraint Systems

The ccp model is parametric in a *constraint system (cs)* specifying the structure and interdependencies of the information that processes can ask or and add to a *central shared store*. This information is represented as assertions traditionally called *constraints*.

Following [10,16] 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* $\sqcup$ is the join of information.

Definition 1 (Constraint Systems). *A constraint system (cs)* $\mathbf{C}$ *is a complete algebraic lattice* $(Con, Con_0, \sqsubseteq, \sqcup, true, false)$ *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.*

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

In order to model *hiding* of local variables and *parameter passing*, in [25,26] the notion of constraint system is enriched with *cylindrification operators* and *diagonal elements*, concepts borrowed from the theory of cylindric algebras [20].

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

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

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

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

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

We assume notions of *free variable* and of *substitution* that satisfy the following conditions, where $c[y/x]$ is the constraint obtained by substituting x by y in c and $fv(c)$ is the set of free variables of c: (1) if $y \notin fv(c)$ then $(c[y/x])[x/y] = c$; (2) $(c \sqcup d)[y/x] = c[y/x] \sqcup d[y/x]$; (3) $x \notin fv(c[y/x])$; (4) $fv(c \sqcup d) = fv(c) \cup fv(d)$.

We now illustrate a constraint system for linear-order arithmetic.

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

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

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

A variable assignment is a function $\mu : Var \longrightarrow \omega$. We use $\mathcal{A}$ to denote the set of all assignments; $\mathcal{P}(X)$ to denote the powerset of a set X, $\emptyset$ the empty set and $\cap$ the

intersection of sets. We use $\mathcal{M}(\phi)$ to denote the set of all assignments that *satisfy* the formula ϕ, where the definition of *satisfaction* is as expected.

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

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

Assumption 1 *We shall assume that the constraint system is well-founded and, for practical reasons, that its ordering $\sqsubseteq$ is decidable. Well-foundedness is needed for technical reasons in the definition of the labeled transition semantics in Section 3.2.*

2.2 Syntax of CCP

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

$$P, Q, \ldots ::= \mathbf{tell}(c) \mid \sum_{i \in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i \mid P \parallel Q \mid \exists_x P \mid p(\boldsymbol{z})$$

where I is a finite set of indexes and $c, c_i \in Con_0$. We use $Proc$ to denote the set of all processes.

Finite processes. Intuitively, the tell process $\mathbf{tell}(c)$ adds c to the global store. The addition is performed regardless the generation of inconsistent information. The process $P \parallel Q$ stands for the *parallel execution* of P and Q.

The guarded-choice $\sum_{i \in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i$ where I is a finite set of indexes, represents a process that can *nondeterministically* choose one of the P_j (with $j \in I$) whose corresponding guard constraint c_j is entailed by the store. The chosen alternative, if any, precludes the others. We shall often write $\mathbf{ask}\ (c_{i_1})\ \rightarrow\ P_{i_1} + \ldots + \mathbf{ask}\ (c_{i_n})\ \rightarrow\ P_{i_n}$ if $I = \{i_1, \ldots, i_n\}$. If no ambiguity arises, we shall omit the "$\mathbf{ask}(c)\ \rightarrow$" when $c = \mathit{true}$. The *blind-choice* process $\sum_{i \in I} \mathbf{ask}\ (\mathit{true})\ \rightarrow\ P_i$, for example, can be written $\sum_{i \in I} P_i$. We shall omit the "$\sum_{i \in I}$" when I is a singleton. We use **stop** as an abbreviation of the empty summation $\sum_{i \in \emptyset} P_i$.

$\exists_x$ is a *hiding operator*, namely it indicates that in $\exists_x P$ the variable x is *local* to P. The occurrences of x in $\exists_x P$ are said to be bound. The bound variables of P, $bv(P)$, are those with a bound occurrence in P, and its free variables, $fv(P)$, are those with an unbound occurrence[3].

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

Infinite processes. To specify infinite behavior, ccp provides parametric process definitions. A process $p(\boldsymbol{z})$ is said to be a *procedure call* with identifier p and actual parameters $\boldsymbol{z}$. We presuppose that for each procedure call $p(z_1 \dots z_m)$ there exists a unique *procedure definition* possibly *recursive*, of the form $p(x_1 \dots x_m) \stackrel{\mathtt{def}}{=} P$ where $fv(P) \subseteq \{x_1, \dots, x_m\}$. Furthermore we require recursion to be *guarded*: I.e., each procedure call within P must occur within an ask process. The behavior of $p(z_1 \dots z_m)$ is that of $P[z_1 \dots z_m / x_1 \dots x_m]$, i.e., P with each x_i replaced with z_i (applying α-conversion to avoid clashes). We shall use $\mathcal{D}$ to denote the set of all process definitions.

Remark 1 (Choice-free fragment of ccp). Henceforth, we use ccp$\backslash$+ to refer to the fragment of ccp without nondeterministic choice. More precisely ccp$\backslash$+ processes are those in which every occurrence of $\sum_{i \in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i$ has its index set I of cardinality 0 or 1.

2.3 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 given above. 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 configurations whose processes are in the ccp$\backslash$+ fragment.

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. Rules R1 and R2 are easily seen to realize the intuitions described in Section 2.2. Rule R3 states that $\sum_{i \in I}\ \mathbf{ask}\ (c_i)\ \rightarrow\ P_i$ can evolve to P_j whenever the global store d entails c_j and $j \in I$.

Rule R4 is somewhat more involved, first we extend the syntax by introducing a process $\exists_x^e P$ representing the evolution of a process of the form $\exists_x P$, where e is the local information (*local store*) produced during this evolution. The process $\exists_x P$ can be seen as a particular case of $\exists_x^e P$: it represents the situation in which the local store is empty. Namely, $\exists_x P = \exists_x^{true} P$.

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

$$\textsf{R1}\ \langle \mathbf{tell}(c), d\rangle \longrightarrow \langle \mathbf{stop}, d \sqcup c\rangle \qquad \textsf{R2}\ \frac{\langle P, d\rangle \longrightarrow \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \longrightarrow \langle P' \parallel Q, d'\rangle}$$

$$\textsf{R3}\ \frac{j \in I \text{ and } c_j \sqsubseteq d}{\langle \sum_{i\in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i, d\rangle \longrightarrow \langle P_j, d\rangle} \qquad \textsf{R4}\ \frac{\langle P, e \sqcup \exists_x d\rangle \longrightarrow \langle P', e' \sqcup \exists_x d\rangle}{\langle \exists_x^e P, d\rangle \longrightarrow \langle \exists_x^{e'} P', d \sqcup \exists_x e'\rangle}$$

$$\textsf{R5}\ \frac{\langle P[\boldsymbol{z}/\boldsymbol{x}], d\rangle \longrightarrow \gamma'}{\langle p(\boldsymbol{z}), d\rangle \longrightarrow \gamma'} \text{ where } p(\boldsymbol{x}) \stackrel{\text{def}}{=} P \text{ is a process definition in } \mathcal{D}$$

Table 1. Reduction semantics for ccp (symmetric rule for R2 is omitted). $\mathcal{D}$ is the set of process definitions.

that the process is transformed into $\exists_x^{e'} P'$, and that the information $\exists_x e$ present in the global store is transformed into $\exists_x e'$.[4]

2.4 Barbed Semantics

In [2], the authors introduced a *barbed semantics* for ccp. Barbed equivalences have been introduced in [19] 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 $c \in Con_0$ and $c \sqsubseteq d$. Similarly, γ satisfies a weak barb c, written $\gamma \Downarrow_c$, iff there exist γ' s.t. $\gamma \longrightarrow^* \gamma' \downarrow_c$.*

Example 2. Consider the constraint system from Example 1 and let $Vars = \{x\}$. Let $\gamma = \langle \mathbf{ask}\ (x > 10)\ \rightarrow\ \mathbf{tell}(x > 42), x > 10\rangle$. We have $\gamma \downarrow_{x>5}$ since $(x > 5) \sqsubseteq (x > 10)$ and $\gamma \Downarrow_{x>42}$ since $\gamma \longrightarrow \langle \mathbf{tell}(x > 42), x > 10\rangle \longrightarrow \langle \mathbf{stop}, (x > 42)\rangle \downarrow_{x>42}$. □

In this context, the equivalence proposed is the *saturated bisimilarity* [8,6]. 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.

Using this idea, in [2], the authors propose a saturated bisimilarity for ccp where condition (iii) requires the bisimulations to be *upward closed* instead of closing under any process context. A *process context* C is a term with a single hole $\bullet$ such that if we replace $\bullet$ with a process P, we obtain a process term $C[P]$. For example, for the parallel context $C = \bullet \parallel R$ we obtain $C[P] = P \parallel R$.

[4] For more details about the operational semantics we refer the reader to [2].

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 $\langle P, true\rangle \dot{\sim}_{sb} \langle Q, true\rangle$.

We shall prove that the closure condition (iii) is enough to make $\dot{\approx}_{sb}$ a congruence in ccp\+. This means that $P \dot{\approx}_{sb} Q$ implies $C[P] \dot{\approx}_{sb} C[Q]$ for every process context. However, this is not the case for ccp with nondeterministic choice as we shall demonstrate later on.

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

Example 4. Take P and Q as in Example 3. One can check that $P \dot{\approx}_{sb} Q$. First notice that $\langle P, true\rangle \not\longrightarrow$ and also $\langle Q, true\rangle \not\longrightarrow$. Now note that for all e it is the case that both configurations evolve to a γ where $\gamma \Downarrow_e$. Intuitively, none of the configurations adds information to the store and, since $\dot{\approx}_{sb}$ does not care about the silent transitions, then P and Q should be weakly bisimilar. □

Finally, notice that in ccp\+ configurations are *confluent* in the following sense.

Proposition 1 (Confluence [26]). *Let $\gamma \in Conf_{\mathtt{ccp}\backslash+}$. If $\gamma \longrightarrow^* \gamma_1$ and $\gamma \longrightarrow^* \gamma_2$ then there exists γ' such that $\gamma_1 \longrightarrow^* \gamma'$ and $\gamma_2 \longrightarrow^* \gamma'$.*

The proposition above will be a cornerstone for the results we shall obtain in ccp\+.

3 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, a given equivalence $\bowtie$ is said to be a congruence if $P \bowtie Q$ implies $C[P] \bowtie C[Q]$ for every process context C[5]. The congruence issue is fundamental for algebraic as well as practical reasons; one may not be content with having $P \bowtie Q$ equivalent but $R \parallel P \not\bowtie R \parallel Q$. Nevertheless, some of the representative equivalences in concurrency are not congruences. For example, in CCS [17], 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 guarded-summation (see [18]).

We shall see that $\dot{\approx}_{sb}$ is a congruence for ccp$\backslash$+. However, this is not the case in the presence of nondeterministic choice. Moreover, unlike CCS, the problem arises even in the presence of guarded summation/choice. In fact, our counterexample reveals that the problem is intrinsic to ccp.

3.1 Observational Equivalence

In this section we shall introduce the standard notion of observational equivalence ($\sim_o$) [26] for ccp as well as its relation with $\dot{\approx}_{sb}$.

The notion of *fairness* is central to the definition of observational equivalence for ccp. We introduce this notion following [12]. Any derivation of a transition involves an application of R1 or R3. We say that P is *active* in a transition $t = \gamma \longrightarrow \gamma'$ if there exists a derivation of t where rule R1 or R3 is used to produce a transition of the form $\langle P, d\rangle \longrightarrow \gamma''$. Moreover, we say that P is *enabled* in γ if there exists γ' such that P is active in $\gamma \longrightarrow \gamma'$. A computation $\gamma_0 \longrightarrow \gamma_1 \longrightarrow \gamma_2 \longrightarrow \ldots$ is said to be *fair* if for each process enabled in some γ_i there exists $j \geq i$ such that the process is active in $\gamma_j \longrightarrow \gamma_{j+1}$.

Note that a finite fair computation is guaranteed to be *maximal*, namely no outgoing transitions are possible from its last configuration.

The standard notion of observables for ccp are the *results* computed by a process for a given initial store. The result of a computation is defined as the least upper bound of all the stores occurring in the computation, which, due to the monotonic properties of ccp, form an increasing chain. More formally:

Definition 5 (Result). *Given a finite or infinite computation ξ of the form:*

$$\xi = \langle Q_0, d_0\rangle \longrightarrow \langle Q_1, d_1\rangle \longrightarrow \langle Q_2, d_2\rangle \longrightarrow \ldots$$

The result of ξ, denoted by $\mathsf{Result}(\xi)$*, is the constraint* $\bigsqcup_i d_i$.

[5] Recall that 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 $C[P] = R \parallel P$.

Note that for a finite computation the result coincides with the store of the last configuration. Now since ccp\+ is confluent (Proposition 1), the following theorem from [26] states that all the fair computations of a configuration have the same result.

Proposition 2 ([26]). *Let γ be ccp\+ configuration and let ξ_1 and ξ_2 be two computations of γ. If ξ_1 and ξ_2 are fair, then* $\mathsf{Result}(\xi_1) = \mathsf{Result}(\xi_2)$.

Before introducing the notion of observational equivalence we need some notation. Below we define the set of possible computations of a given configuration.

Definition 6 (Set of Computations). *The set of computations starting from γ, denoted* $\mathsf{Comp}(\gamma)$*, is defined as:*

$$\mathsf{Comp}(\gamma) = \{\xi \mid \xi = \gamma \longrightarrow \gamma' \longrightarrow \gamma'' \longrightarrow \ldots\}$$

Now we introduce the notion of observables. Intuitively, the set of observables of γ is the set of results of the fair computations starting from γ.

Definition 7 (Observables). *Let $\mathcal{O} : Proc \to Con_0 \to 2^{Con}$ be given by:*

$$\mathcal{O}(P)(d) = \{e \mid \xi \in \mathsf{Comp}(\langle P, d\rangle), \xi \textit{ is fair and } \mathsf{Result}(\xi) = e\}.$$

Using these elements we define the notion of observational equivalence. Two configurations are deemed equivalent if they have the same set observables for any given store.

Definition 8 (Observational equivalence). *We say that P and Q are observational equivalent, written $P \sim_o Q$, iff $\mathcal{O}(P) = \mathcal{O}(Q)$.*

Notice that in the case of ccp\+, as defined in [26], the set of observables is a singleton because of Proposition 2.

Remark 2. Let $\langle P, d\rangle \in Conf_{\mathtt{ccp}\backslash+}$. Note that $\mathcal{O} : Proc \to Con_0 \to Con$ because of Proposition 2 and it is defined as $\mathcal{O}(P)(d) = \mathsf{Result}(\xi)$ where ξ is any fair computation of $\langle P, d\rangle$.

In [2] it was shown that, in ccp\+, weak saturated barbed bisimilarity and observation equivalence coincide. Recall that $P \dot{\approx}_{sb} Q$ means $\langle P, true\rangle \dot{\approx}_{sb} \langle Q, true\rangle$.

Proposition 3 ([2]). *Let P and Q be ccp\+ processes. Then $P \sim_o Q$ iff $P \dot{\approx}_{sb} Q$.*

Nevertheless, the above theorem does not hold for ccpwith nondeterministic choice. We can show this by using a counter-example reminiscent from the standard one for CCS. Let $P = (\mathbf{ask}\ (b) \ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (b) \ \rightarrow\ \mathbf{tell}(d))$ and $Q = \mathbf{ask}\ (b) \ \rightarrow\ ((\mathbf{ask}\ (true) \ \rightarrow\ \mathbf{tell}(c)) + (\mathbf{ask}\ (true) \ \rightarrow\ \mathbf{tell}(d)))$. One can verify that $P \sim_o Q$ but $P \not\dot{\approx}_{sb} Q$. However, the ($\Leftarrow$) direction of the theorem does hold as we show next.

Theorem 1. *If $P \dot{\approx}_{sb} Q$ then $P \sim_o Q$.*

$$\mathsf{LR1}\ \langle \mathbf{tell}(c), d\rangle \xrightarrow{true} \langle \mathbf{stop}, d \sqcup c\rangle \qquad \mathsf{LR2}\ \frac{\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \xrightarrow{\alpha} \langle P' \parallel Q, d'\rangle}$$

$$\mathsf{LR3}\ \frac{j \in I \text{ and } \alpha \in \min\{a \in Con_0 \mid c_j \sqsubseteq d \sqcup a\}}{\langle \sum_{i\in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i, d\rangle \xrightarrow{\alpha} \langle P_j, d \sqcup \alpha\rangle}$$

$$\mathsf{LR4}\ \frac{\langle P[z/x], e[z/x] \sqcup d\rangle \xrightarrow{\alpha} \langle P', e' \sqcup d \sqcup \alpha\rangle}{\langle \exists^{e}_{x} P, d\rangle \xrightarrow{\alpha} \langle \exists^{e'[x/z]}_{x} P'[x/z], \exists_x (e'[x/z]) \sqcup d \sqcup \alpha\rangle}$$

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

$$\mathsf{LR5}\ \frac{\langle P[\boldsymbol{z}/\boldsymbol{x}], d\rangle \xrightarrow{\alpha} \gamma'}{\langle p(\boldsymbol{z}), d\rangle \xrightarrow{\alpha} \gamma'} \text{ where } p(\boldsymbol{x}) \stackrel{\text{def}}{=} P \text{ is a process definition in } \mathcal{D}$$

Table 2. Labeled semantics for ccp (symmetric rule for LR2 is omitted).

3.2 Congruence

We begin this section by showing that weak bisimilarity is a congruence in a restricted sense: It is preserved by all the contexts from the choice-free fragment. For this purpose it is convenient to recall the labeled semantics of ccp as well as the (labeled) weak bisimilarity introduced in [2].

Labeled Semantics In a labeled 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 $\xrightarrow{true}$.

The LTS $(Conf, Con_0, \longrightarrow)$ is defined by the rules in Table 2. The rule LR3, for example, says that $\langle \sum_{i\in I} \mathbf{ask}\ (c_i)\ \rightarrow\ P_i, d\rangle$ can evolve to $\langle P_j, d \sqcup \alpha\rangle$ if $j \in I$ and the environment provides a minimal constraint α that added to the store d entails the guard c_j, i.e., $\alpha \in \min\{a \in Con_0 \mid c_j \sqsubseteq d \sqcup a\}$. Notice that Assumption 1 guarantees the existence of α. The rule LR4 follows the same approach as R4, however it uses variable substitution instead of hiding with the existential operator.[6] The other rules are easily seen to realize the intuition given in Section 2.2.

We can now introduce the notion of weak bisimilarity ($\dot{\approx}$) from [2]. In [2] it is shown that $\dot{\approx}$ coincides with $\dot{\approx}_{sb}$ and, by exploiting the labeled semantics, avoids the upward closure from condition (iii) in $\dot{\approx}_{sb}$.

[6] See [2] for a detailed explanation of the rule LR4.

Definition 9 (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 $\langle P, true\rangle \dot{\approx} \langle Q, true\rangle$.

To illustrate this definition consider the following example.

Example 5. 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 $\gamma_1 \dot{\approx} \gamma_2$ when $d \sqsubseteq c$. Intuitively, this corresponds to the fact that the implication $c \Rightarrow d$ is equivalent to $true$ when c already 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 bisimulation as in Definition 9. □

The following result from [2] states that weak bisimilarity coincides with weak saturated barbed bisimilarity (Definition 4).

Proposition 4 ([2]). $\dot{\approx}_{sb} = \dot{\approx}$.

We can now prove that $\dot{\approx}_{sb}$ is a congruence in ccp\+.

Theorem 2. *Let P and Q be ccp\+ processes and assume that $P \dot{\approx}_{sb} Q$. Then for every process context $C[\bullet]$ in ccp\+ we have $C[P] \dot{\approx}_{sb} C[Q]$.*

Notice that this result implies that observational equivalence ($\sim_o$) is a congruence. Unfortunately the theorem above does not hold for ccp with nondeterministic choice, as shown next.

Theorem 3. *There exists P', Q, R in ccp s.t. (a) $P' \dot{\approx}_{sb} Q$ but (b) $P' \parallel R \not\dot{\approx}_{sb} Q \parallel R$.*

Proof. 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 1. It can easily be verified that (the symmetric closure of) $\mathcal{R}$ is a weak bisimulation (see Definition 9). The point (a) then follows from Proposition 4.

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 pair $(\langle P \parallel R, e\rangle, \langle Q \parallel R, true\rangle)$. The results then follows from Proposition 4 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:

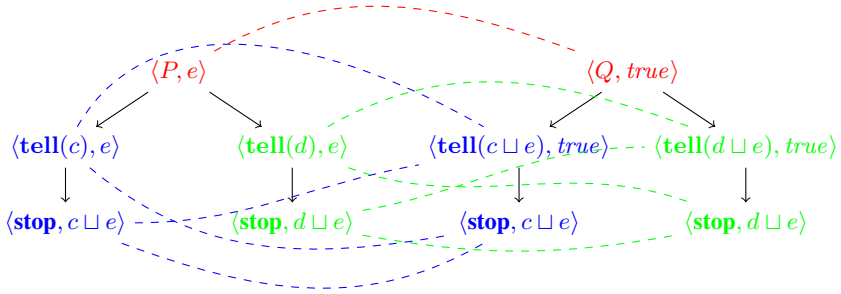

Fig. 1. Let $P = (\mathbf{ask}\,(true) \rightarrow \mathbf{tell}(c)) + (\mathbf{ask}\,(true) \rightarrow \mathbf{tell}(d))$ and $Q = (\mathbf{ask}\,(true) \rightarrow \mathbf{tell}(c \sqcup e)) + (\mathbf{ask}\,(true) \rightarrow \mathbf{tell}(d \sqcup e))$. The linked configurations are weakly bisimilar.

- $\langle Q \parallel R, true\rangle$ *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.*

4 Weak full bisimilarity

In the previous section we showed that $\dot{\approx}$ (and $\dot{\approx}_{sb}$) for the full ccp is not entirely satisfactory since it is not a congruence. By building on $\dot{\approx}$, in this section we propose a new equivalence which we call *(weak) full bisimilarity*, written $\approx_f$. This new equivalence does not quantify over infinitely many process contexts in its definition yet we will show that is a congruence. Furthermore, we will also prove that adequacy of $\approx_f$ by showing that it is the largest congruence included in $\dot{\approx}_{sb}$.

4.1 More than weak barbs

The key to figure out the element missing in the definition of $\dot{\approx}_{sb}$ (Definition 4) lies in Figure 1. If we look at the configurations in the figure we can see that while $\langle P, e\rangle$ is able to *produce* a barb e without choosing between c and d, $\langle Q, true\rangle$ is not. The definition of $\dot{\approx}_{sb}$ tries to capture this in the condition (i), namely by checking that $\langle P, e\rangle \Downarrow_e$ then requiring that $\langle Q, true\rangle \Downarrow_e$. However, this condition does not capture the fact that in

order to produce e, $\langle Q, \mathit{true}\rangle$ may have to evolve into a configuration which can no longer produce some of the weak barbs $\langle Q, \mathit{true}\rangle$ can produce. [7]

Using this insight, we shall define a new notion of weak bisimilarity that changes condition (i) in $\dot{\approx}$ (Definition 9) in order to deal with the problem present in Figure 1. More concretely, condition (i) requires that whenever $\langle P, c\rangle \downarrow_\alpha$ then $\langle Q, d\rangle \Downarrow_\alpha$, $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle \downarrow_\alpha$ without imposing any condition between $\langle P, c\rangle$ and $\langle Q', d'\rangle$. This makes it possible that $\langle P, c\rangle \downarrow_\beta$ and $\langle Q', d'\rangle$ does *not*: indeed, it might be the case that that $\langle Q, d\rangle \longrightarrow^* \langle Q'', d''\rangle \downarrow_\beta$ for some other branch $\langle Q'', d''\rangle$. Hence $\langle P, c\rangle$ and $\langle Q, d\rangle$ would pass condition (i) as in Figure 1.

Weak full bisimilarity deals with this problem by adding a condition between $\langle P, c\rangle$ and $\langle Q', d'\rangle$, namely $\langle Q, d\rangle \Downarrow_c$ has to hold by reaching a bisimilar configuration: $\langle P, c\rangle$ has to be weakly bisimilar $\langle Q', d'\rangle$.

Definition 10 (Weak Full Bisimilarity). *A weak full 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) there exists $\gamma_2' = \langle Q', d'\rangle$ such that $\langle Q, d\rangle \longrightarrow^* \gamma_2'$ where $c \sqsubseteq d'$ and $(\gamma_1, \gamma_2') \in \mathcal{R}$,

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

We say that γ_1 and γ_2 are weak fully bisimilar ($\gamma_1 \approx_f \gamma_2$) if there exists a weak full bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$. We write $P \approx_f Q$ iff $\langle P, \mathit{true}\rangle \approx_f \langle Q, \mathit{true}\rangle$.

In the definition above, the fist condition states that $\langle Q, d\rangle$ has to produce c by reaching a (weakly) bisimilar configuration. The second condition is the bisimulation game from $\dot{\approx}$ (Definition 9) plus a condition requiring the store c' to be matched too.

To better explain this notion consider again the counterexample to $\dot{\approx}$ from Figure 1.

Example 6. Let $\langle P, e\rangle, \langle Q, \mathit{true}\rangle$ as in Figure 1. Let us build a relation $\mathcal{R}$ that is a weak full bisimulation where $(\langle P, e\rangle, \langle Q, \mathit{true}\rangle) \in \mathcal{R}$. By condition (i) in Definition 10 we need a $\gamma_2' = \langle Q', d'\rangle$ s.t. $\langle Q, d\rangle \longrightarrow^* \gamma_2'$ and $e \sqsubseteq d'$ and $(\gamma_1, \gamma_2') \in \mathcal{R}$. We have two options $Q' = \mathbf{stop}$ and $d' = c \sqcup e$ or $d' = d \sqcup e$.[8] However, if we take $(\langle P, e\rangle, \langle \mathbf{stop}, c \sqcup e\rangle) \in \mathcal{R}$ we have that $\langle P, e\rangle \Downarrow_d$ while $\langle \mathbf{stop}, c \sqcup e\rangle \not\Downarrow_d$. A similar argument works for $\langle \mathbf{stop}, d \sqcup e\rangle$. Therefore, no weak full bisimulation may contain $(\langle P, e\rangle, \langle Q, \mathit{true}\rangle)$. Hence $\langle P, e\rangle \not\approx_f \langle Q, \mathit{true}\rangle$. □

4.2 Congruence issues

We shall now prove that full bisimilarity is a congruence w.r.t all possible contexts in ccp. Namely, whenever γ and γ' are in $\approx_f$ then they can be replaced for one another in any context.

[7] In the case of ccp\+ this is not a concern given that in this fragment weak barbs are always preserved during evolution.

[8] The cases for $Q' = \mathbf{tell}(c \sqcup e)$ or $Q' = \mathbf{tell}(d \sqcup e)$ with $d' = \mathit{true}$ are equivalent.

Theorem 4. *Let P and Q be ccp processes and assume that $P \approx_f Q$. Then for every process context $C[\bullet]$ we have that $C[P] \approx_f C[Q]$.*

Proof. Here we consider the parallel case; the other cases are trivial or easier to verify. We shall prove that $\mathcal{R} = \{(\langle P \parallel R, c\rangle, \langle Q \parallel R, d\rangle) \mid \langle P, c\rangle \approx_f \langle Q, d\rangle\}$ is a weak full bisimulation as in Definition 10.

*To prove (i), since $\langle P, c\rangle \approx_f \langle Q, d\rangle$ we have that $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle$ where $c \sqsubseteq d'$ and $\langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$ **(1)**. Therefore by* R2 *we get $\langle Q \parallel R, d\rangle \longrightarrow^* \langle Q' \parallel R, d'\rangle$ and by* **(1)** *we can conclude that $(\langle Q' \parallel R, d'\rangle, \langle P \parallel R, c\rangle) \in \mathcal{R}$.*

To prove (ii) let us assume that $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c_1\rangle$. We proceed by induction (on the depth) of the inference of $\langle P \parallel R, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle$.

Using LR2 *(left), then $P_1 = (P' \parallel R)$ with $\langle P, c\rangle \xrightarrow{\alpha} \langle P', c'\rangle$ by a shorter inference. Since $\langle P, c\rangle \approx_f \langle Q, d\rangle$ then $\langle Q, d \sqcup \alpha\rangle \longrightarrow^* \langle Q', d'\rangle$ where $\langle P', c'\rangle \approx_f \langle Q', d'\rangle$ and $c' \sqsubseteq d'$ **(3)**. By* R2 *we have $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d'\rangle$ and from* **(3)** *we can conclude that $(\langle P' \parallel R, c\rangle, \langle Q' \parallel R, d'\rangle) \in \mathcal{R}$.*

Using LR2 *(right), then $P_1 = (P \parallel R')$ and $c' = (c \sqcup \alpha \sqcup e)$ with $\langle R, c\rangle \xrightarrow{\alpha} \langle R', c'\rangle$ by a shorter inference. From* **(1)** *we know that $\langle Q, d\rangle \longrightarrow^* \langle Q', d'\rangle$ where $c \sqsubseteq d'$ and $\langle Q', d'\rangle \dot{\approx} \langle P, c\rangle$. Hence $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R, d' \sqcup \alpha\rangle$. Now since $c \sqsubseteq d'$ then by monotonicity $\langle R, d' \sqcup \alpha\rangle \longrightarrow \langle R, d''\rangle$ where $d'' = d' \sqcup \alpha \sqcup e$. Therefore by* R2 *we get $\langle Q \parallel R, d \sqcup \alpha\rangle \longrightarrow^* \langle Q' \parallel R', d''\rangle$ and from* **(1)** *and monotonicity $\langle P, c'\rangle = \langle P, c \sqcup \alpha \sqcup e\rangle \dot{\approx} \langle Q', d' \sqcup \alpha \sqcup e\rangle = \langle Q', d''\rangle$. Using this we can conclude that $(\langle P \parallel R', c'\rangle, \langle Q' \parallel R', d''\rangle) \in \mathcal{R}$.*

It is clear that $\approx_f$ is more distinguishing than $\dot{\approx}$ and the result above shows that this level of granularity is needed if we want a weak bisimilarity that is a congruence for the full ccp.

4.3 Relation with observational equivalence

In section 3.1 we described the relation between weak (saturated) bisimilarity ($\dot{\approx}_{sb}$, Definition 4) and the standard observational equivalence ($\sim_o$, Definition 8) for ccp. Concretely, we know that, in ccp$\backslash$+, $\dot{\approx}_{sb}$ coincides with $\sim_o$, while for the full ccp $\dot{\approx}_{sb}$ implies $\sim_o$ but the converse does not hold. In this section we shall see the relation between weak full bisimilarity ($\approx_f$, Definition 10) and $\sim_o$. We shall prove that $\approx_f$ coincides with $\sim_o$ in ccp$\backslash$+ by proving that $\approx_f$ corresponds to $\dot{\approx}_{sb}$ in the choice-free fragment of ccp. Furthermore, for the full language of ccp, we shall prove that $\approx_f$ implies $\sim_o$ again by showing that $\approx_f$ implies $\dot{\approx}_{sb}$ in ccp.

Let us start by showing that $\approx_f$ and $\dot{\approx}$ coincide in ccp$\backslash$+. This theorem strongly relies on the confluent nature of ccp$\backslash$+ (Proposition 1).

Theorem 5. *Let $\gamma, \gamma' \in Conf_{\texttt{ccp}\backslash+}$, $\gamma \approx_f \gamma'$ iff $\gamma \dot{\approx} \gamma'$.*

The corollary below follows from Proposition 3 and 4, and Theorem 5.

Corollary 1. *Let P and Q be ccp$\backslash$+ processes. Then $P \approx_f Q$ iff $P \sim_o Q$.*

We shall now prove that $\approx_f$ implies $\sim_o$ for the full ccp. In order to do this we first prove that $\approx_f$ implies $\dot{\approx}_{sb}$.

Theorem 6. *If* $\gamma \approx_f \gamma'$ *then* $\gamma \dot{\approx} \gamma'$.

The corollary below follows from Theorem 1 and 6, and Proposition 4.

Corollary 2. *If* $P \approx_f Q$ *then* $P \sim_o Q$.

The above statement allows us to use the co-inductive techniques of full bisimulation to prove observational equivalence.

4.4 Behavioral congruence

Finally, we prove that $\approx_f$ is the largest congruence included in $\dot{\approx}$ by showing that it coincides with the congruence $\dot{\cong}$ defined next.

Definition 11 (Behavioral Congruence). *We say that* P *is behaviorally congruent to* Q*, denoted* $P \dot{\cong} Q$*, iff for every process context* $C[\bullet]$ *we have* $C[P] \dot{\approx} C[Q]$. *We use* $\langle P, e\rangle \dot{\cong} \langle Q, d\rangle$ *to denote* $(P \parallel \mathbf{tell}(e)) \dot{\cong} (Q \parallel \mathbf{tell}(d))$.

We now state that $\approx_f$ coincides with $\dot{\cong}$ for ccp with nondeterministic choice.

Theorem 7. $\langle P, e\rangle \approx_f \langle Q, d\rangle$ *iff* $\langle P, e\rangle \dot{\cong} \langle Q, d\rangle$.

5 Conclusions and Related Work

In this paper we showed that the weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$) proposed in [2] is not a congruence for ccp. Nevertheless, we also showed that the upward closure, i.e. condition (iii), is enough to make $\dot{\approx}_{sb}$ a congruence in the choice-free fragment (ccp\+). We then proposed a new notion of bisimilarity, called weak full bisimilarity ($\approx_f$), and we proved that it is a congruence for the full ccp despite the fact that $\approx_f$ does not require any quantification over a (potentially) infinite number of contexts in its definition. Furthermore, we showed that $\approx_f$ implies the standard observational equivalence ($\sim_o$) for ccp from [26]. Finally we demonstrated that $\approx_f$ is not too restrictive by showing that it is the largest congruence included in $\dot{\approx}_{sb}$. See Table 3 for a summary of the contributions of this paper. This is the first weak behavioral ccp congruence for ccp with nondeterministic choice that does not require implicit quantification over all contexts.

Most of the related work has already been discussed in the introduction (Section 1). There has been other attempts for finding a good notion of bisimilarity for ccp such as [25] and [16]. In [25] the authors propose a ccp bisimilarity that requires processes to match the exact label in the bisimulation game, a condition which is standard in process calculi realm, however this notion is known to be too distinguishing for ccp as shown in [2]. As for [16], their notion of (strong) bisimilarity resembles to the saturated barbed bisimilarity from [2] and, although they do not give a notion of weak bisimilarity, the results in this paper can be related directly.

We plan to adapt the algorithms from [3,23] to verify $\approx_f$. We conjecture that the decision procedure for $\dot{\approx}_{sb}$ can be exploited to check $\approx_f$ by modifying the way the (weak) barbs are considered. Furthermore, in this paper we obtained a notion of weak

Language	Relation among equivalences	Congruence w.r.t.	
		$C[\bullet]$	$C[\bullet]\backslash+$
ccp\+	$\dot{\cong} = \approx_f = \dot{\approx} = \dot{\approx}_{sb} = \sim_o$	N/A	$\dot{\cong}, \approx_f, \dot{\approx}, \dot{\approx}_{sb}, \sim_o$
ccp	$\dot{\cong} = \approx_f \subseteq \dot{\approx} = \dot{\approx}_{sb} \subseteq \sim_o$	$\dot{\cong}, \approx_f$	$\dot{\cong}, \approx_f, \dot{\approx}, \dot{\approx}_{sb}$

Table 3. Summary of the contributions. Recall that $\dot{\approx}_{sb}$ stands for the weak saturated barbed bisimilarity (Definition 4), $\sim_o$ is the standard observational equivalence (Definition 8), $\dot{\approx}$ represents weak bisimilarity (Definition 9), $\approx_f$ is the notion of weak full bisimilarity proposed in this paper (Definition 10) and $\dot{\cong}$ stands for the behavioral congruence (Definition 11). $C[\bullet]\backslash+$ stands for the contexts where the summation operator does not occur, while $C[\bullet]$ represents any possible context, hence the summation operator may occur in $C[\bullet]$. For this reason we put N/A (Not Applicable) in the row corresponding to ccp\+. Notice that the correspondence $\dot{\approx} = \dot{\approx}_{sb} = \sim_o$ comes from [2].

bisimilarity that is a congruence even if we do not consider a label for observing the tell actions. Since ccp is an asynchronous language, not observing the tell follows the philosophy of considering as labels the minimal information needed to proceed, namely a tell process does not need a stimulus from the environment to post its information in the store. Following the same reasoning, we plan to investigate whether it is possible to define a labeled semantics for the asynchronous π-calculus ($A\pi$) [18,24] with a τ label for the output transitions, instead of a co-action, and we shall check if a notion of bisimilarity similar to ours would also be a congruence.

References

1. F. Arbab and J. J. M. M. Rutten. A coinductive calculus of component connectors. In *WADT*, pages 34–55, 2002.
2. A. Aristizábal, F. Bonchi, C. Palamidessi, L. Pino, and F. D. Valencia. Deriving labels and bisimilarity for concurrent constraint programming. In M. Hofmann, editor, *14th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2011)*, volume 6604 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2011.
3. A. Aristizábal, F. Bonchi, L. Pino, and F. D. Valencia. Partition refinement for bisimilarity in CCP. In S. Ossowski and P. Lecca, editors, *27th Annual ACM Symposium on Applied Computing (SAC 2012)*, pages 88–93. ACM, 2012.
4. M. Bartoletti and R. Zunino. A calculus of contracting processes. In *25th Annual IEEE Symposium on Logic in Computer Science (LICS 2010)*, pages 332–341. IEEE Computer Society, 2010.
5. J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, pages 39–48. IEEE Computer Society, 2009.
6. F. Bonchi, F. Gadducci, and G. V. Monreale. Reactive systems, barbed semantics, and the mobile ambients. In L. de Alfaro, editor, *12th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2009)*, volume 5504 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2009.

7. F. Bonchi, F. Gadducci, and G. V. Monreale. Towards a general theory of barbs, contexts and labels. In H. Yang, editor, *APLAS*, volume 7078 of *Lecture Notes in Computer Science*, pages 289–304. Springer, 2011.
8. F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 69–80. IEEE Computer Society, 2006.
9. M. G. Buscemi and U. Montanari. Open bisimulation for the concurrent constraint pi-calculus. In S. Drossopoulou, editor, *17th European Symposium on Programming Languages and Systems (ESOP 2008)*, volume 4960 of *Lecture Notes in Computer Science*, pages 254–268. Springer, 2008.
10. F. S. de Boer, A. D. Pierro, and C. Palamidessi. Nondeterminism and infinite computations in constraint programming. *Theoretical Computer Science*, 151(1):37–78, 1995.
11. R. De Nicola. Behavioral equivalences. In *Encyclopedia of Parallel Computing*, pages 120–127. 2011.
12. M. Falaschi, M. Gabbrielli, K. Marriott, and C. Palamidessi. Confluence in concurrent constraint programming. *Theoretical Computer Science*, 183(2):281–315, 1997.
13. W. Fokkink, J. Pang, and A. Wijs. Is timed branching bisimilarity a congruence indeed? *Fundam. Inform.*, 87(3-4):287–311, 2008.
14. S. Knight, C. Palamidessi, P. Panangaden, and F. D. Valencia. Spatial and epistemic modalities in constraint-based process calculi. In M. Koutny and I. Ulidowski, editors, *23rd International Conference on Concurrency Theory (CONCUR 2012)*, volume 7454 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2012.
15. J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In C. Palamidessi, editor, *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 243–258. Springer, 2000.
16. N. P. Mendler, P. Panangaden, P. J. Scott, and R. A. G. Seely. A logical view of concurrent constraint programming. *Nordic Journal of Computing*, 2(2):181–220, 1995.
17. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.
18. R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.
19. R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *19th International Colloquium on Automata, Languages and Programming (ICALP 1992)*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer, 1992.
20. J. Monk, L. Henkin, and A. Tarski. *Cylindric Algebras (Part I)*. North-Holland, 1971.
21. C. Palamidessi, V. A. Saraswat, F. D. Valencia, and 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)*, pages 59–68. IEEE Computer Society, 2006.
22. L. F. Pino, F. Bonchi, and F. D. Valencia. A behavioral congruence for concurrent constraint programming with nondeterministic choice (extended version). Technical report, INRIA/DGA and LIX, École Polytechnique, France, 2013. `http://www.lix.polytechnique.fr/~luis.pino/files/ictac14-extended.pdf`.
23. L. F. Pino, F. Bonchi, and F. D. Valencia. Efficient computation of program equivalence for confluent concurrent constraint programming. In R. Peña and T. Schrijvers, editors, *15th International Symposium on Principles and Practice of Declarative Programming (PPDP 2013)*, pages 263–274. ACM, 2013.
24. D. Sangiorgi and D. Walker. *The π-Calculus - a theory of mobile processes*. Cambridge University Press, 2001.
25. V. A. Saraswat and M. C. Rinard. Concurrent constraint programming. In F. E. Allen, editor, *17th Annual ACM Symposium on Principles of Programming Languages (POPL 1991)*, pages 232–245. ACM Press, 1990.

26. V. A. Saraswat, M. C. Rinard, and P. Panangaden. Semantic foundations of concurrent constraint programming. In D. S. Wise, editor, *18th Annual ACM Symposium on Principles of Programming Languages (POPL 1991)*, pages 333–352. ACM Press, 1991.