Observational and Behavioural Equivalences for Soft Concurrent Constraint Programming

Fabio Gadducci[a,*], Francesco Santini[b,1,**], Luis F. Pino[c], Frank D. Valencia[d]

[a]*Dipartimento di Informatica, Università di Pisa, Italy*
[b]*Dipartimento di Matematica e Informatica, Università di Perugia, Italy*
[c]*Dipartimento di Matematica e Informatica, Università di Cagliari, Italy*
[d]*CNRS and LIX, École Polytechnique de Paris, France*

Abstract

We present a labelled semantics for Soft Concurrent Constraint Programming (SCCP), a meta-language where concurrent agents may synchronise on a shared store by either posting or checking the satisfaction of (soft) constraints. SCCP generalises the classical formalism by parametrising the constraint system over an order-enriched monoid, thus abstractly representing the store with an element of the monoid, and the standard unlabelled semantics just observes store updates. The novel operational rules are shown to offer a sound and complete co-inductive technique to prove the original equivalence over the unlabelled semantics. Based on this characterisation, we provide an axiomatisation for finite agents.

1. Introduction

Concurrent Constraint Programming (CCP) [1] is a language based on a shared-memory communication pattern: processes may interact by either posting or checking partial information, which is represented as constraints in a global store. CCP belongs to the larger family of process calculi, thus a syntax-driven operational semantics represents the computational steps.

*Principal corresponding author.
**Secondary corresponding author.
Email addresses: gadducci@di.unipi.it (Fabio Gadducci), francesco.santini@dmi.unipg.it (Francesco Santini), luis.pino@unica.it (Luis F. Pino), frank.valencia@lix.polytechnique.fr (Frank D. Valencia)

For example, the term **tell**(c) represents a process that posts c in the store, and the term **ask**(c) $\rightarrow$ P is the process that executes P if c can be derived from the information in the store.

The formalism is parametric with respect to the entailment relation. Under the name of *constraint system*, the information recorded on the store is structured as a partial order (in fact, a lattice) $\leq$, where $c \leq d$ means that c can be derived from d. Under a few requirements over such systems, CCP has been provided with (coincident) operational and denotational semantics. Recently, a labelled semantics has also been provided, and the associated weak bisimilarity proved to coincide with the original semantics [2].

A key aspect of CCP is the *idempotency* of the operator for composing constraints: adding the same information twice does not change the store. On the contrary, the soft variant of the formalism (Soft CCP, or just SCCP [3]) drops idempotency: constraint systems in SCCP may distinguish the number of occurrences of a piece of information. Dropping idempotency requires a complete reworking of the theory. Although an operational semantics for SCCP has been devised [3], hitherto neither the denotational nor the labelled one has been reintroduced. This is unfortunate since due to its generality, suitable SCCP instances has been successfully applied as a specification formalism for negotiation of Service Level Agreements [4], or the enforcement of ACL-based access control [5, 6].

As a language, SCCP has been used as a specification formalism for agents collaborating via a shared knowledge basis, possibly with temporal features [7, 8]. Thus, on a methodological level, the development of behavioural equivalences for SCCP may result in the improvement on the analysis techniques for agents that need to reason guided by their preferences, more so if their knowledge is not complete.

In more general terms, SCCP represents, by its parametric nature, a formal meta-model where to develop different constraint-languages over different (weighted or crisp) logics, as it will clearly appear from Section **??**.

The work in [1] establishes a denotational semantics for CCP and an equational theory for infinite agents. More recently, in [2] the authors prove that the axiomatisation is underlying a specific weak bisimilarity among agents, thus providing a clear operational understanding. The key ingredients are a complete lattice as the domain of the store, with least upper bound for constraint combination, and a notion of compactness such that domain equations for the parallel composition of recursive agents would be well-defined. On the contrary, the soft version [3] drops the upper

bound for combination in exchange of a more general monoidal operator. Thus, the domain is potentially just a (not necessarily complete) partial order, possibly with finite meets and a residuation operator (a kind of inverse of the monoidal one) in order to account for algorithms concerning constraint propagation. Indeed, the main use of SCCP has been in the generalisation of classical constraint satisfaction problems, hence the lack of investigation about e.g compactness and denotational semantics.

The objective of our work is the development of a general theory for the operational semantics of SCCP, via the introduction of suitable observational and behavioural equivalences. Reaching this objective is technically challenging, since most of the simplicity of CCP is based precisely on the premise that posting an information multiple times is the same as posting it only once. The first step consists in recasting the notion of compactness from crisp to soft; we then introduce a novel labelled semantics for SCCP which will allow us to give a sound and complete technique to prove the equivalence over the unlabelled semantics.

We will build our framework by supposing to have a global store, that is shared by all the agents. Such a premise is required by how we design the transition system: in fact, it is labelled with a shared set of variables (i.e., Δ) as a means to keep track of variables' name after renaming them through the hiding operator. Since variables support the constraints posted to the store, renaming is more subtle than in other algebras: the store retains a memory of former names. In the future we plan to investigate how hiding can be performed in case of stores local to each agent (see Sec. 7).

This paper details the work in [9] and extends it by reconnecting the presented framework with the classical work on soft constraint systems [3] (see Section 5). Section 2 opens the paper with the technical background, presenting also some novelty as $\otimes$-compact elements. Section **??** presents the semantics of a deterministic fragment of a constraint language, together with fundamental concepts, e.g., observables, confluence, observational equivalence, and, in Section **??**, the notion of saturated bisimulation. Section 3 derives a labelled transition system for SCCP, soundness and completeness with respect to the unlabelled one, and weak/strong bisimilarity relations. Section 4 shows a sound and complete axiomatisation for a finite fragment of the language. Section 5 presents a case study concerning classical SCCP [3], while Section 6 presents some related examples. Finally, Section 7 wraps up the paper with conclusions and future work.

2. The Algebraic Background

This section recalls the main notions we are going to need later on. First of all, we present some basic facts concerning monoids [10] enriched over complete lattices. These are used to recast the standard presentation of the soft constraints paradigm, and to generalise the classical crisp one.

2.1. Lattice-enriched Monoids

Definition 1 (Complete lattices). *A partial order (PO) is a pair $\langle A, \leq \rangle$ such that A is a set of values and $\leq \subseteq A \times A$ is a reflexive, transitive, and anti-symmetric relation. A complete lattice (CL) is a PO such that any subset of A has a least upper bound (LUB).*

We denote as $\bigvee X$ the necessarily unique LUB of a subset $X \subseteq A$, and explicitly $\bot$ and $\top$ if we are considering the empty set and the whole A, respectively: the former is the bottom and the latter is the top of the PO. Obviously, CLs also have the greatest lower bound (GLB) for any subset $Y \subseteq A$, denoted as $\bigwedge Y$. In the following we fix a CL $\mathbb{C} = \langle A, \leq \rangle$.

Definition 2 (Compact elements). *An element $a \in A$ is compact (or finite) if whenever $a \leq \bigvee Y$ there exists a finite subset $X \subseteq Y$ such that $a \leq \bigvee X$.*

Note that for complete lattices the definition of compactness given above coincides with the one using directed subsets. It will be easier to generalise it, though, to compactness with respect to the monoidal operator (see Definition 6). We let $A^C \subseteq A$ denote the set of compact elements of $\mathbb{C}$.

Example 1. *Note that A^C might be trivial. Consider e.g. the CL $\langle [0,1], \geq \rangle$ (the segment of the reals with the inverse of the usual order), used for probabilistic constraints [11]: only the bottom element 1 is compact. As we will see, the situation for the soft paradigm is more nuanced.*

Definition 3 (Monoids). *A commutative monoid with identity (IM) is a triple $\langle A, \otimes, \mathbf{1} \rangle$ such that $\otimes : A \times A \to A$ is a commutative and associative function and*

(identity) $\forall a \in A. \otimes (a, \mathbf{1}) = a$

We will often use an infix notation, such as $a \otimes b$ for $a, b \in A$. The monoidal operator can be defined for any multi-set: it is given for a family of elements $a_i \in A$ indexed over a finite, non-empty set I, and it is denoted by $\bigotimes_{i \in I} a_i$. If for an index set I the a_i's are different, we write $\bigotimes S$ instead of $\bigotimes_{i \in I} a_i$ for the set $S = \{a_i \mid i \in I\}$. Conventionally, we denote $\bigotimes \emptyset = \bot$.

We now move our attention to our choice for the domain of values.

Definition 4 (CL-enriched IMs). *A CL-enriched IM (CLIM) is a triple* $\mathbb{S} = \langle A, \leq, \otimes \rangle$ *such that* $\langle A, \leq \rangle$ *is a CL,* $\langle A, \otimes, \bot \rangle$ *is an IM, and*

(distributivity) $\forall a \in A. \forall X \subseteq A. a \otimes \bigwedge X = \bigwedge \{a \otimes x \mid x \in X\}$

Remark 1. *The reader who is familiar with the soft constraint literature may have noticed that we have basically rewritten the standard presentation using a CLIM instead of an absorptive semiring, recently popularised as c-semiring [12], where the* $a \oplus b$ *operator is replaced by the binary LUB* $a \vee b$*. Besides what we consider a streamlined presentation, the main advantage in the use of CLIMs is the easiness in defining the LUB of infinite sets and, as a consequence, the notion of* $\otimes$*-compactness given below. An alternative solution using infinite sums can be found in [13, Section 3], and a possible use is sketched in [14].*

Thanks to distributivity, we can show that $\otimes$ is monotone, and since $\bot$ is the identity of the monoid, monotonicity implies that the combination of constraints is increasing, i.e., $\forall a, b \in A. a \leq a \otimes b$ holds. Finally, we recall that by definition $\bigwedge \emptyset = \top$, so that $\forall a \in A. a \otimes \top = \top$ also holds.[2]

In the following, we fix a CLIM $\mathbb{S} = \langle A, \leq, \otimes \rangle$. The next step is to provide an infinite composition. Our definition is from [10] (see also [13, p. 42]).

Definition 5 (Infinite composition). *Let I be a (countable) set of indexes. Then, the composition* $\bigotimes_{i \in I} a_i$ *is defined as* $\bigvee_{J \subseteq I} \bigotimes_{j \in J} a_j$ *for all finite subsets J.*

Should I be finite, the definition gives back the usual multi-set composition, since $\otimes$ is monotone and increasing. As the infinitary composition is also monotone and increasing, by construction $\bigotimes A = \bigvee A = \top$. We now provide a notion of compactness with respect to the monoidal operator.

[2] A symmetric choice $\langle A, \otimes, \top \rangle$ with distributivity with respect to $\bigvee$ (and thus $a \otimes \bot = \bot$) is possible: the monoidal operator would be decreasing, so that for example $a \otimes b \leq a$. Indeed, this is the usual order in the semiring-based approach to soft constraints [14].

Definition 6 (⊗-compact elements). *Let $a \in A$. It is ⊗-compact (or ⊗-finite) if whenever $a \leq \bigotimes_{i \in I} a_i$ then there exists a finite subset $J \subseteq I$ such that $a \leq \bigotimes_{j \in J} a_j$.*

We let $A^{\otimes} \subseteq A$ denote the set of ⊗-compact elements of $\mathbb{S}$. It is easy to show that a compact element is also ⊗-compact, i.e. $A^{C} \subseteq A^{\otimes}$. Indeed, the latter notion is definitively more flexible.

Example 2. *Consider the CLIM $\langle [0,1], \geq, \times \rangle$ examined above, which corresponds to the segment of the reals with the inverse of the usual order and multiplication as monoidal product. Since any infinite multiplication tends to 0, then all the elements are ⊗-compact, except the top element itself, that is, precisely 0.*

Remark 2. *It is easy to show that idempotency implies that $\bigotimes$ coincides with LUBs, that is, $\bigotimes S = \bigvee S$ for all subsets $S \subseteq A$. In other words, the whole soft structure collapses to a complete distributive lattice. Indeed, requiring distributivity makes the soft paradigm not fully comparable with the crisp one. We are going to discuss it again in the concluding remarks.*

2.2. Residuation

Our first operator on CL-enriched IMs is a simple construction for a weak inverse of the monoidal operator in CL-enriched monoids, known in the literature on enriched monoid as residuation [13, 15].

Definition 7 (Residuation). *Let $a, b \in A$. The residuation of a with respect to b is defined as $a \oslash b = \bigwedge \{c \in A \mid a \leq b \otimes c\}$.*

The definition conveys the intuitive meaning of a division operator: indeed, $a \leq b \otimes (a \oslash b)$, thanks to distributivity. Also, $(a \otimes b) \oslash b \leq a$ and $a \oslash (b \otimes c) = (a \oslash b) \oslash c$. Residuation is monotone on the first argument: if $a \leq b$ then $a \oslash c \leq b \oslash c$ and $a \oslash b = \bot$. For more properties of residuation we refer to [16, Table 4.1].

Most important for our formalism is the following result on ⊗-compactness.

Lemma 1. *Let $a, b \in A$. If a is ⊗-compact, so is $a \oslash b$.*

Proof 1. *If $a \oslash b \leq \bigotimes_{i \in I} a_i$, then by monotonicity $a \leq \bigotimes_{i \in I \uplus \{*\}} a_i$ for $a_* = b$. By ⊗-compactness of a there exists a finite $J \subseteq I$ such that $a \leq \bigotimes_{j \in J \uplus \{*\}} a_j$, and by the definition of division $a \oslash b \leq \bigotimes_{j \in J} a_j$, hence the result holds.*

Most standard soft instances (boolean, fuzzy, probabilistic, weighted, and so on) are described by CL-enriched monoids and are residuated: see e.g. [14]. For these instances the $\oplusminus$ operator is used to (partially) remove constraints from the store, and as such is going to be used in Section 3. In fact, in the soft literature it is usually required a tighter relation of (full) invertibility, also satisfied by all the previous CLIMs instances, stated in our framework by the definition below.

Definition 8. *$\mathbb{S}$ is invertible if $b \leq a$ implies $b \otimes (a \oplusminus b) = a$ for all $a, b \in A^{\otimes}$.*

2.3. Cylindrification

We now consider two families of operators for modellng the hiding of variables and the passing of parameters in soft CCP. They can be considered as generalised notions of existential quantifier and diagonal element [1], which are expressed in terms of operators of cylindric algebras [17]. [3]

Definition 9 (Cylindrification). *Let V be a set of variables. A cylindric operator $\exists$ over $\mathbb{S}$ and V is a family of monotone, $\otimes$-compactness preserving functions $\exists_x : A \to A$ indexed by elements in V such that for all $a, b \in A$ and $x, y \in V$*

1. $\exists_x a \leq a$;
2. $\exists_x(a \otimes \exists_x b) = \exists_x a \otimes \exists_x b$;
3. $\exists_x \exists_y a = \exists_y \exists_x a$.

Let $a \in A$. The support *of a is the set of variables $sv(a) = \{x \in V \mid \exists_x a \neq a\}$.*

For a finite $X \subseteq V$, let $\exists_X a$ denote any sequence of function applications. We now fix a set of variables V and a cylindric operator $\exists$ over $\mathbb{S}$ and V.

Definition 10 (Diagonalisation). *A diagonal operator δ for $\exists$ is a family of idempotent elements $\delta_{x,y} \in A$ indexed by pairs of elements in V such that $\delta_{x,y} = \delta_{y,x}$ and for all $a \in A$ and $x, y, z \in V$*

1. $\delta_{x,x} = \bot$;

[3] However, since we consider monoids instead of groups, the set of axiom of diagonal operators is included in the standard one for cylindric algebras.

2. *if* $z \notin \{x, y\}$ *then* $\delta_{x,y} = \exists_z(\delta_{x,z} \otimes \delta_{z,y})$;

3. *if* $x \neq y$ *then* $a \leq \delta_{x,y} \otimes \exists_x(a \otimes \delta_{x,y})$.

Axioms 1 and 2 above plus idempotency imply that $\exists_x \delta_{x,y} = \bot$, which in turn implies (by axiom 2 and idempotency of $\exists$) that $sv(\delta_{x,y}) = \{x, y\}$ for $x \neq y$. Diagonal operators are used for modeling variable substitution and parameter passing. In the following, we fix a diagonal operator δ for $\exists$.

Definition 11 (Substitution). *Let* $x, y \in V$ *and* $a \in A$. *The substitution* $a[^y/_x]$ *is defined as* a *if* $x = y$ *and as* $\exists_x(\delta_{x,y} \otimes a)$ *otherwise.*

Substitution behaves correctly with respect to $\exists$.

Lemma 2. *Let* $x, y, w \in V$ *and* $a \in A$. *Then it holds*

- $(\exists_x a)[^y/_x] = a$
- $\exists_x a = \exists_y(a[^y/_x])$ *for* $y \notin sv(a)$
- $(\exists_w a)[^y/_x] = \exists_w(a[^y/_x])$ *if* $w \notin \{x, y\}$

Finally, we rephrase some further laws of the crisp case (see [2, p.140]).

Lemma 3. *Let* $x, y \in V$ *and* $a \in A$. *Then it holds*

1. $(a[^y/_x])[^x/_y] = a$ *for* $y \notin sv(a)$
2. $a[^y/_x] \otimes b[^y/_x] = (a \otimes b)[^y/_x]$
3. $x \notin sv(a[^y/_x])$.

Proof 2. *Consider e.g. the most difficult item* 2. *By definition* $a[^y/_x] \otimes b[^y/_x] = \exists_x(\delta_{x,y} \otimes a) \otimes \exists_x(\delta_{x,y} \otimes b)$, *which in turn coincides with* $\exists_x(\delta_{x,y} \otimes a \otimes \exists_x(\delta_{x,y} \otimes b))$ *by axiom* 2 *of* $\exists$; *by axiom* 3 *of* $\delta_{x,y}$ *we have that* $(a \otimes b)[^y/_x] = \exists_x(\delta_{x,y} \otimes a \otimes b) \leq \exists_x(\delta_{x,y} \otimes a \otimes \exists_x(\delta_{x,y} \otimes b))$, *while the vice versa holds by the monotonicity of* $\exists_x$.

LA1	$\dfrac{sv(\sigma) \cup sv(c) \subseteq \Delta}{\langle \mathbf{tell}(c), \sigma \rangle \overset{\perp}{\longmapsto}_{\Delta} \langle \mathbf{stop}, \sigma \otimes c \rangle}$	**Tell**
LA2	$\dfrac{sv(\sigma) \cup sv(c) \cup fv(A) \subseteq \Delta}{\langle \mathbf{ask}(c) \rightarrow A, \sigma \rangle \overset{c \oplus \sigma}{\longmapsto}_{\Delta} \langle A, \sigma \otimes (c \oplus \sigma) \rangle}$	**Ask**
LA3	$\dfrac{sv(\sigma) \cup \{y\} \subseteq \Delta \wedge p(x) = A \in \mathcal{P}}{\langle p(y), \sigma \rangle \overset{\perp}{\mapsto}_{\Delta} \langle A[^{y}/_{x}], \sigma \rangle}$	**Rec**
LA4	$\dfrac{sv(\sigma) \cup fv(\exists_x A) \subseteq \Delta \wedge w \notin \Delta}{\langle \exists_x A, \sigma \rangle \overset{\perp}{\mapsto}_{\Delta} \langle A[^{w}/_{x}], \sigma \rangle}$	**Hide**

Table 1: Axioms of the labelled semantics for SCCP.

3. A Labelled Transition System for Soft CCP

Although $\approx_s$ is fully abstract, it is somewhat unsatisfactory because of the upward-closure, i.e., the quantification in condition *3* of Definition **??**.

Definition 12 (Labelled reductions). *Let* $\Gamma = \mathcal{A} \times C^{\otimes}$ *be the set of* configurations *and* V *the set of variables. The* labelled direct reduction semantics *for SCCP is the pair* $\langle \Gamma, \mapsto \rangle$ *such that* $\mapsto \subseteq \Gamma \times \Gamma$ *is indexed over the couple* $\langle C^{\otimes}, 2^{V} \rangle$, *i.e.,* $\mapsto = \bigcup_{\alpha \in C^{\otimes}, \Delta \subseteq V} \overset{\alpha}{\mapsto}_{\Delta}$ *and* $\overset{\alpha}{\mapsto}_{\Delta} \subseteq \Gamma \times \Gamma$, *obtained by the rules in Table 1.*

The labelled reduction semantics *for SCCP is the pair* $\langle \Gamma, \rightarrow \rangle$ *such that* $\rightarrow \subseteq \Gamma \times \Gamma$ *is the family of binary relations indexed over the couple* $\langle C^{\otimes}, 2^{V} \rangle$; $\overset{\alpha}{\rightarrow}_{\Delta} \subseteq \Gamma \times \Gamma$ *can obtained by the rules in Table 1 and Table 2.*

In Table 1 and Table 2 we refine the notion of transition (respectively given in Table **??** and Table **??**) by adding a label that carries additional information about the constraints that cause the reduction. Hence, we define a new labelled transition system (LTS) obtained by the family of relations $\overset{\alpha}{\mapsto}_{\Delta} \subseteq \Gamma \times \Gamma$ indexed over $\langle C^{\otimes}, 2^{V} \rangle$; as a reminder, Γ is the set of configurations, $C^{\otimes}$ the set of $\otimes$-compact constraints, and, as for the unlabelled semantics in Section **??**, transitions are indexed by sets of variables. Rules in Table 1 and Table 2 are identical to those in Table **??** and Table **??**, except for a constraint α that represents the minimal information that must be added to σ in order to fire an action from $\langle A, \sigma \rangle$ to $\langle A', \sigma' \rangle$, i.e., $\langle A, \sigma \otimes \alpha \rangle \longrightarrow_{\Delta} \langle A', \sigma' \rangle$.

$$\textbf{LR1}\quad \frac{\langle A,\sigma\rangle \xrightarrow{\alpha}_{\Delta} \langle A',\sigma'\rangle \wedge fv(B)\subseteq\Delta}{\langle A \parallel B,\sigma\rangle \xrightarrow{\alpha}_{\Delta} \langle A' \parallel B,\sigma'\rangle}\quad \textbf{Par1}$$

$$\textbf{LR2}\quad \frac{\langle A,\sigma\rangle \xrightarrow{\alpha}_{\Delta} \langle A',\sigma'\rangle \wedge fv(B)\subseteq\Delta}{\langle B \parallel A,\sigma\rangle \xrightarrow{\alpha}_{\Delta} \langle B \parallel A',\sigma'\rangle}\quad \textbf{Par2}$$

Table 2: Contextual rules of the labelled semantics for SCCP.

Rule **LA2** says that $\langle \mathbf{ask}(c) \to A, \sigma\rangle$ can evolve to $\langle A, \sigma \otimes \alpha\rangle$ if the environment provides a minimal constraint α that added to the store σ entails c, i.e., $\alpha = c \oplusminus \sigma$. Notice that, differently from [2], here the definition of this minimal label comes directly from a derived operator of the underlying CLIM (i.e., from $\oplusminus$), which by Lemma 1 preserves $\otimes$-compactness.

The LTS is sound and complete with respect to the unlabelled semantics.

Lemma 4 (Soundness). *Let A be an agent and $\sigma \in C^{\otimes}$. If $\langle A,\sigma\rangle \overset{\alpha}{\mapsto}_{\Delta} \langle A',\sigma'\rangle$ then $\langle A, \sigma\otimes\alpha\rangle \mapsto_{\Delta} \langle A',\sigma'\rangle$.*

Proof 3. *We just consider* **LA2***: The other cases are straightforward.*

*By **LA2** we have $A = \mathbf{ask}(c) \to A'$, $\alpha = c \oplusminus \sigma$, and $\sigma' = (\sigma \otimes (c \oplusminus \sigma))$. Since it always holds that $c \leq (\sigma \otimes (c \oplusminus \sigma))$ then by **A2** we have $\langle A, \sigma\otimes\alpha\rangle \mapsto_{\Delta} \langle A',\sigma'\rangle$.*

We are going to need a stronger notion of completeness. Intuitively, whenever there exists a direct reduction $\langle A,\sigma\rangle \mapsto$, then a corresponding labelled direct reduction exists originating from any $\langle A,\rho\rangle$ such that $\rho \leq \sigma$.

Lemma 5 (Completeness). *Let A be an agent and $\sigma, \rho \in C^{\otimes}$ such that $\rho \leq \sigma$. Moreover, let us assume C to be invertible. If $\langle A,\sigma\rangle \mapsto_{\Delta} \langle A',\sigma'\rangle$ then there exist $\alpha, a \in C^{\otimes}$ such that $\langle A,\rho\rangle \overset{\alpha}{\mapsto}_{\Delta} \langle A',\rho'\rangle$ with $\rho\otimes\alpha\otimes a = \sigma$ and $\rho'\otimes a = \sigma'$.*

Proof 4. *We just consider* **LA2***: The other cases are straightforward and they are verified by always choosing $\alpha = a = \bot$.*

*By **LA2** we have $A = \mathbf{ask}(c) \to A'$, $\sigma' = \sigma$, and $c \leq \sigma$. Now consider $\langle A,\rho\rangle \overset{\alpha}{\mapsto}_{\Delta} \langle A',\rho'\rangle$, where $\alpha = c \oplusminus \rho$ and $\rho' = \rho \otimes \alpha$. Also, let us note that $c \leq \sigma$ and $\rho \leq \sigma$ imply $\rho' \leq \sigma$ by invertibility. Thus, we can take $a = \sigma \oplusminus \rho'$, and the conditions are easily verified again by the invertibility of C.*

The complex statement boils down to the checking for the satisfaction of a constraint c in a store ρ that does not necessarily verify it. This is the equivalent of [2, Lemma 5] for the crisp language, and it is going to be needed in Lemma 6 of the next section to prove that weak and strong bisimilarities are upward closed with respect to the store.

The lemmata above can in any case be simplified to state that the labelled reduction semantics is in fact an extension of the unlabelled one.

Proposition 1. *Let A be an agent and $\sigma \in \mathcal{C}^{\otimes}$. Then $\langle A, \sigma\rangle \xrightarrow{\perp}_{\Delta} \langle A', \sigma'\rangle$ if and only if $\langle A, \sigma\rangle \longrightarrow_{\Delta} \langle A', \sigma'\rangle$.*

3.1. Strong and Weak Bisimilarity on the LTS.

We now define an equivalence that characterises $\sim_s$ without the upward closure condition. As it occurs with the crisp language, and differently from calculi such as Milner's CCS, barbs cannot be removed from the definition of bisimilarity because they cannot be inferred by the transitions.

Definition 13 (Strong bisimilarity). *A strong bisimulation is a symmetric relation R on configurations such that if $(\gamma_1, \gamma_2) \in R$ for $\gamma_1 = \langle A, \sigma\rangle$ and $\gamma_2 = \langle B, \rho\rangle$*

1. *if $\gamma_1 \downarrow_c$ then $\gamma_2 \downarrow_c$*
2. *if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ then $\exists \gamma_2'$ such that $\langle B, \rho \otimes \alpha\rangle \longrightarrow \gamma_2'$ and $(\gamma_1', \gamma_2') \in R$.*

We say that γ_1 and γ_2 are strongly bisimilar ($\gamma_1 \sim \gamma_2$) if there exists a strong bisimulation R such that $(\gamma_1, \gamma_2) \in R$.

Whenever σ and ρ are $\otimes$-compact elements, the first condition is equivalent to require $\sigma \leq \rho$. Thus $(\gamma_1, \gamma_2) \in R$ would imply that γ_1 and γ_2 have the same store. As for the second condition, we adopted a *semi-saturated* equivalence, introduced for CCP in [2]. In the bisimulation game a label can be simulated by a reduction including in the store the label itself.

Definition 14 (Weak bisimilarity). *A weak bisimulation is a symmetric relation R on configurations such that if $(\gamma_1, \gamma_2) \in R$ for $\gamma_1 = \langle A, \sigma\rangle$ and $\gamma_2 = \langle B, \rho\rangle$*

1. *if $\gamma_1 \downarrow_c$ then $\gamma_2 \Downarrow_c$*
2. *if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ then $\exists \gamma_2'$ such that $\langle B, \rho \otimes \alpha\rangle \longrightarrow^* \gamma_2'$ and $(\gamma_1', \gamma_2') \in R$.*

We say that γ_1 and γ_2 are weakly bisimilar ($\gamma_1 \approx \gamma_2$) if there exists a weak bisimulation R such that $(\gamma_1, \gamma_2) \in R$.

With respect to the weak equivalence for crisp constraints, some of its characteristic equalities do not hold, e.g. $\mathbf{ask}(c) \rightarrow \mathbf{tell}(c) \not\approx \mathbf{stop}$. As usual, this is due to the fact that the underlying CLIM may not be idempotent.

We can now conclude by proving the equivalence between $\sim_s$ and $\sim$ and between $\approx_s$ and $\approx$ (hence, $\approx$ is further equivalent to $\sim_o$, using Proposition **??**). We start by showing that $\sim$ is preserved under closure.

Lemma 6. *Let $\langle A, \sigma\rangle$, $\langle B, \rho\rangle$ be configurations and $c \in C^{\otimes}$ Moreover, let C be invertible. If $\langle A, \sigma\rangle \sim \langle B, \rho\rangle$, then $\langle A, \sigma \otimes c\rangle \sim \langle B, \rho \otimes c\rangle$.*

Proof 5. *We need to show that $R = \{(\langle A, \sigma \otimes a\rangle \sim \langle B, \rho \otimes a\rangle) \mid \langle A, \sigma\rangle \sim \langle B, \rho\rangle\}$ satisfies the two properties in Definition 13.*

i) From the hypothesis $\langle A, \sigma\rangle \sim \langle B, \rho\rangle$, we have that $\rho = \sigma$, thus $\langle A, \sigma \otimes a\rangle$ and $\langle B, \rho \otimes a\rangle$ satisfy the same barbs.

ii) Let us assume $\langle A, \sigma \otimes c\rangle \xrightarrow{\alpha} \langle A', \sigma'\rangle$, we need to prove the existence of B' and ρ' such that $\langle B, \rho \otimes c \otimes \alpha\rangle \rightarrow \langle B', \rho'\rangle$ and $(\langle A', \sigma'\rangle, \langle B', \rho'\rangle) \in R$. By Lemma 4 and Lemma 5 we obtain $\langle A, \sigma\rangle \xrightarrow{\alpha'} \langle A', \sigma''\rangle$ and the existence of a' such that $\sigma \otimes \alpha' \otimes a' = \sigma \otimes c \otimes \alpha$ (1) and $\sigma'' \otimes a' = \sigma'$ (2). By invertibility, we also get that (1') $\alpha' \otimes a' = c \otimes \alpha$. From the labelled transition of $\langle A, \sigma\rangle$ and the hypothesis $\langle A, \sigma\rangle \sim \langle B, \rho\rangle$, we have that $\langle B, \rho \otimes \alpha'\rangle \rightarrow \langle B', \rho''\rangle$, with $\langle A, \sigma''\rangle \sim \langle B, \rho''\rangle$ (3). By (1') we have $\langle B, \rho \otimes c \otimes \alpha\rangle = \langle B, \rho \otimes \alpha' \otimes a'\rangle$ and $\langle B, \rho \otimes \alpha' \otimes a'\rangle \rightarrow \langle B, \rho'' \otimes a'\rangle$ (due to operational monotonicity). Finally, by the definition of R and (3), we conclude that $(\langle A', \sigma'' \otimes a'\rangle, \langle B', \rho'' \otimes a'\rangle) \in R$, and, by (2), $\langle A', \sigma'' \otimes a'\rangle = \langle A', \sigma'\rangle$.

Theorem 1. *$\sim_s \subseteq \sim \subseteq$. Moreover, let C be invertible. Then $\sim_s = \sim$*

Proof 6. *The inclusion $\sim \subseteq \sim_s$ can be proved by using Lemma 6.*

From $\sim$ to $\sim_s$. *We show that $R = \{(\langle A, \sigma\rangle, \langle B, \rho\rangle) \mid \langle A, \sigma\rangle \sim \langle B, \rho\rangle\}$ is a saturated bisimulation, i.e., if $(\langle A, \sigma\rangle, \langle B, \rho\rangle) \in R$ the conditions in Definition **??** hold.*

i) If $\langle A, \sigma\rangle \downarrow_c$, then we have $\langle B, \rho\rangle \downarrow_c$ by the hypothesis $\langle A, \sigma\rangle \sim \langle B, \rho\rangle$.

ii) Suppose that $\langle A, \sigma\rangle \to \langle A', \sigma'\rangle$. *By Proposition 1 we have* $\langle A, \sigma\rangle \xrightarrow{\perp} \langle A', \sigma'\rangle$. *Since* $\langle A, \sigma\rangle \sim \langle B, \rho\rangle$, *then* $\langle B, \rho \otimes \perp\rangle \to \langle B', \rho'\rangle$ *with* $\langle A', \sigma'\rangle \sim \langle B', \rho'\rangle$. *Since* $\rho = \rho \otimes \perp$, *we have* $\langle B, \rho\rangle \to \langle B', \rho'\rangle$.

iii) By Lemma 6, $(\langle A, \sigma \otimes c'\rangle, \langle B, \rho \otimes c'\rangle) \in R$ *for all* $c' \in C^{\otimes}$.

From $\sim_s$ **to** $\sim$. *We show that* $R = \{(\langle A, \sigma\rangle, \langle B, \rho\rangle) \mid \langle A, \sigma\rangle \sim_s \langle B, \rho\rangle\}$ *is a strong bisimulation, i.e., if* $(\langle A, \sigma\rangle, \langle B, \rho\rangle) \in R$ *the conditions in Definition 13 hold.*

i) If $\langle A, \sigma\rangle \downarrow_c$, *then we have* $\langle B, \rho\rangle \downarrow_c$ *by the hypothesis* $\langle A, \sigma\rangle \sim_s \langle B, \rho\rangle$.

ii) Suppose that $\langle A, \sigma\rangle \xrightarrow{\alpha} \langle A', \sigma'\rangle$. *Then by Lemma 4 we have* $\langle A, \sigma \otimes \alpha\rangle \to \langle A', \sigma'\rangle$. *Since* $\langle A, \sigma\rangle \sim_s \langle B, \rho\rangle$, *then* $\langle A, \sigma \otimes \alpha\rangle \sim_s \langle B, \rho \otimes \alpha\rangle$ *and thus* $\langle B, \rho \otimes \alpha\rangle \to \langle B', \rho'\rangle$ *with* $\langle A', \sigma'\rangle \sim_s \langle B', \rho'\rangle$.

To prove the correspondence between weak bisimulations, we need a result similar to Lemma 6. The key is the preservation of weak barbs by the addition of constraints to the store, which is trivial in the strong case.

Lemma 7. *Let* $\langle A, \sigma\rangle$, $\langle B, \rho\rangle$ *be configurations and* $c, d \in C^{\otimes}$. *If* $\langle A, \sigma\rangle \approx \langle B, \rho\rangle$ *and* $\langle A, \sigma \otimes d\rangle \downarrow_c$, *then* $\langle B, \rho \otimes d\rangle \Downarrow_c$.

Proof 7. *If* $\langle A, \sigma \otimes a\rangle \downarrow_c$, *then* $c \leq \sigma \otimes d$. *Since* $\langle A, \sigma\rangle \approx \langle B, \rho\rangle$ *and* $\langle A, \sigma\rangle \downarrow_\sigma$, *then there exists* $\langle B', \rho'\rangle$ *such that* $\langle B, \rho\rangle \to^* \langle B', \rho'\rangle$ *and* $\sigma \leq \exists_\Gamma \rho'$ *for* $\Gamma = fv(\langle B', \rho'\rangle) \setminus fv(\langle B, \rho\rangle)$. *Let us assume, without loss of generality, that* $\Gamma \cap sv(d) = \emptyset$; *since reductions are monotone (item 4 of Lemma* **??***), we have* $\langle B, \rho \otimes d\rangle \to^* \langle B', \rho' \otimes d\rangle$. *Finally,* $c \leq \sigma \otimes d = \sigma \otimes \exists_\Gamma d \leq \exists_\Gamma \rho' \otimes \exists_\Gamma d \leq \exists_\Gamma(\rho' \otimes d)$, *hence* $\langle B, \rho \otimes d\rangle \Downarrow_c$.

The result below uses Lemma 7 and rephrases the proof of Lemma 6.

Lemma 8. *Let* $\langle A, \sigma\rangle$, $\langle B, \rho\rangle$ *be configurations and* $c \in C^{\otimes}$ *Moreover, let us assume* C *to be invertible. If* $\langle A, \sigma\rangle \approx \langle B, \rho\rangle$, *then* $\langle A, \sigma \otimes c\rangle \approx \langle B, \rho \otimes c\rangle$.

Now the theorem below is proved by precisely mimicking the proof for Theorem and using Lemma 8 instead of Lemma 6.

Theorem 2. $\approx_s \subseteq \approx \subseteq$. *Moreover, let* C *be invertible. Then* $\approx_s = \approx$.

3.2. Labelled versus saturated semantics.

The main appeal of saturated semantics resides in being a congruence and, in fact, the minimal congruence contained in standard bisimulation [18]. The main drawback of this approach is that it is in principle necessary to check the behaviour of a process under every context. The problem is somewhat mitigated for SCCP, since it suffices to close the store with respect to any possible compact element (item 3 of Definition **??**). At the same time, checking the feasibility of a reduction may require some computational effort, either for solving the combinatorial problem associated with calculating $\sigma \otimes d$, or for verifying if $c \leq \sigma$, as with agent $\mathbf{ask}(c) \rightarrow A$.

This is the reason for searching labelled semantics and suitable notions of bisimilarity that may alleviate such a burden. The intuition is to consider labels which somehow represent the "minimal context allowing a process to reduce", so that a bisimilarity-checking algorithm in principle needs to verify this minimal context only, instead of every one. The idea has been exploited in the framework of crisp CCP [2], and it is based on [19, 20].

Example 3. *Let us consider the agents* $\mathbf{ask}(c) \rightarrow \mathbf{stop}$ *and* $\mathbf{stop}$*. To prove that they are weakly bisimilar, it has to be proved that* $\gamma \stackrel{\cdot}{\approx} \gamma'$ *for configurations* $\gamma = \langle \mathbf{ask}(c) \rightarrow \mathbf{stop}, \bot \rangle$ *and* $\gamma' = \langle \mathbf{stop}, \bot \rangle$*. Consider the following relation*

$$\mathcal{R} = \{(\langle \mathbf{ask}(c) \rightarrow \mathbf{stop}, \bot \rangle, \langle \mathbf{stop}, \bot \rangle), (\langle \mathbf{stop}, c \rangle, \langle \mathbf{stop}, c \rangle)\}$$

It is quite easy to prove that it is a bisimulation, and in fact the smallest one identifying the two configurations. It suffices to note that by definition $c \ominus \bot = c$*.*

In order to prove that $\gamma \stackrel{\cdot}{\approx}_s \gamma'$*, instead, we surely need to consider an infinite relation. Indeed, the smallest saturated bisimulation equating the two configuration is given by the relation below*

$$\mathcal{S} = \{(\langle \mathbf{ask}(c) \rightarrow \mathbf{stop}, d \rangle, \langle \mathbf{stop}, d \rangle), (\langle \mathbf{stop}, e \rangle, \langle \mathbf{stop}, e \rangle) \mid d, e \in C^{\otimes} \,\&\, c \leq e\}$$

The relation above clearly is a saturated bisimulation, but any naive automatic check for that property might involve rather complex calculations.

Another reason for the complexity of checking saturated bisimilarity is the need of considering the closure $\longrightarrow^*$ of the reduction relation, which may cause a combinatorial explosion. Think e.g. of the agents $\prod_{i \in I} \mathbf{ask}(c_i) \rightarrow \mathbf{stop}$ and $\mathbf{stop}$. Of course, they might be proved equivalent by exploiting the fact that saturated bisimilarity is a congruence, and by

verifying that **stop** $\parallel A \mathrel{\dot{\approx}_s} A$ for all the agents A. A direct proof would instead require a check for each store of the reductions arising from all the possible interleaving of the c_i elements.

4. An Axiomatisation of Simple Agents for Weak Bisimilarity

Once the behaviour of an agent is captured by an observational equivalence, it is natural to look for laws characterising it. Given its correspondence with the standard equivalence via fair computations, weak bisimilarity is the preferred behavioural semantics for soft CCP. A sound and complete axiomatisation was proposed for CCP in [1]. However, the lack of idempotence in the soft formalism makes unsound some of the axioms there, since posting a constraint twice is different from adding it just once. As an example, law *L1* of [1] states that $\mathbf{ask}(c) \rightarrow \mathbf{tell}(d) = \mathbf{ask}(c) \rightarrow (\mathbf{tell}(c) \parallel \mathbf{tell}(d))$, which is clearly false in the soft formalism.

Nevertheless, most of the axioms in [1] can be recovered, as long as we mix together the various operators available in a CLIM.[4] So, let an agent be simple if it is finite and contains no occurrence of an existential quantifier. Our set of axioms for simple agents of SCCP in presented in Figure 1.

As for sequential processes, in Eqs. 1-2 we present the axioms involving the **0** and in Eqs. 3-4 those related to *ask*. Those concerning the parallel composition operator are instead represented in Eqs. 5-7. Distributivity of *tell* and *ask* are presented in Eqs. 8-9, while Eqs. 10-11 show how further simplify the combination of *tell* and *ask* through parallel composition.

Proposition 2. *The axioms in Figure 1 are sound and complete for simple agents with respect to weak bisimilarity.*

Soundness is easily checked, while completeness is obtained by mimicking the proof schema adopted for the crisp case, exploiting a normal form that is in fact reminiscent of the one in [1, Definition 3.2].[5]

Lemma 9. *Let A be an agent. Then either $A =$ **stop** or it can be decomposed as $\|_i \mathbf{ask}(c_i) \rightarrow \mathbf{tell}(d_i)$ such that*

[4] Our axioms follow closely those of [1, p.341], replacing the law *L1* mentioned above with $\mathbf{tell}(\bot) = \mathbf{stop}$ and dropping altogether the laws *L11* and *L12*.

[5] With respect to the properties stated in [1, p.342], we weakened property 1 and dropped property 4, the latter being linked to axiom L12.

$$\mathbf{tell}(\bot) = \mathbf{stop} \tag{1}$$
$$\mathbf{ask}(c) \to \mathbf{stop} = \mathbf{stop} \tag{2}$$
$$\mathbf{ask}(c) \to \mathbf{ask}(d) \to A = \mathbf{ask}(c \vee d) \to A \tag{3}$$
$$\mathbf{ask}(\bot) \to A = A \tag{4}$$
$$A \parallel B = B \parallel A \tag{5}$$
$$A \parallel (B \parallel C) = (A \parallel B) \parallel C \tag{6}$$
$$A \parallel \mathbf{stop} = A \tag{7}$$
$$\mathbf{tell}(c) \parallel \mathbf{tell}(d) = \mathbf{tell}(c \otimes d) \tag{8}$$
$$\mathbf{ask}(c) \to (A \parallel B) = (\mathbf{ask}(c) \to A) \parallel (\mathbf{ask}(c) \to B) \tag{9}$$
$$\mathbf{ask}(a) \to \mathbf{tell}(b) \parallel \mathbf{ask}(a \otimes b) \to \mathbf{tell}(d) = \mathbf{ask}(a) \to \mathbf{tell}(b \otimes d) \tag{10}$$
$$\mathbf{ask}(a) \to \mathbf{tell}(b) \parallel \mathbf{ask}(c) \to \mathbf{tell}(d) = \mathbf{ask}(a) \to \mathbf{tell}(b) \parallel \mathbf{ask}(c \vee (a \otimes b)) \to \mathbf{tell}(d) \text{ if } a \leq c \tag{11}$$

Figure 1: Axioms for simple agents.

- $d_i \neq \bot$
- $c_i \neq c_j$ *for all* $i \neq j$
- $c_i < c_j$ *implies* $c_i \otimes d_i < c_j$

Axioms 1-8 guarantee that an agent can be decomposed as $\|_i \mathbf{ask}(c_i) \to \mathbf{tell}(d_i)$ such that $d_i \neq \bot$. Axiom 9 then ensures the second condition. Finally, by applying as long as possible first axiom 11 and then axiom 10, the final condition is also enforced.

5. A Case Study and Examples: Soft Constraint Systems

This section recalls the key notions of the soft constraint framework, and cast it into our CLIM formalism (following, yet generalising [3]).

Definition 15 (Constraints). *Let V be a (possibly ordered) set of variables and D a finite domain of interpretation for V. Then, a* constraint *$(V \to D) \to A$ is a function associating a value in A to each assignment $\eta : V \to D$ of the variables.*

We define C as the set of constraints that can be built starting from chosen $\mathbb{S}$, V and D. The application of a constraint function $c : (V \to D) \to A$ to a variable assignment $\eta : V \to D$ is denoted $c\eta$. Note that even if a constraint involves all the variables in V, it may depend on the assignment of a finite subset of them, called its support. For instance,

a binary constraint c with $supp(c) = \{x, y\}$ is a function $c : (V \to D) \to A$ which depends only on the assignment of variables $\{x, y\} \subseteq V$, meaning that two assignments $\eta_1, \eta_2 : V \to D$ differing only for the image of variables $z \notin \{x, y\}$ coincide (i.e., $c\eta_1 = c\eta_2$). The support corresponds to the classical notion of scope of a constraint. We often refer to a constraint with support X as c_X. Moreover, an assignment over a support X of size k is concisely represented by a tuple t in D^k and we often write $c_X(t)$ instead of $c_X\eta$.

The set of constraints (with finite support) clearly forms a CLIM, where the structure is lifted from $\mathbb{S}$. Furthermore, it enjoys the cylindric properties, as shown by the result below (due to [3]).

Proposition 3 (The CLIM of constraints). *The cylindric CLIM of constraints $\mathbb{C}$ is defined as $\langle C, \leq, \otimes \rangle$ such that*

- $c_1 \leq c_2$ *if* $c_1\eta \leq c_2\eta$ *for all* $\eta : V \to D$
- $(c_1 \otimes c_2)\eta = c_1\eta \otimes c_2\eta$ *for all* $c_1, c_2 \in C$
- $(\exists_x c)\eta = \bigwedge_{d \in D} c\eta[x := d]$ *for all* $c \in C, x \in V$
- $\delta_{x,y}\eta = \begin{cases} \bot & \text{if } \eta(x) = \eta(y); \\ \top & \text{otherwise.} \end{cases}$ *for all* $x, y \in V$

Combining two constraints by the $\otimes$ operator means building a new constraint whose support involves at most the variables of the original ones, and which associates with each tuple of domain values for such variables the element which is obtained by multiplying those associated by the original constraints to the appropriate sub-tuples.

Hiding means eliminating variables from the support, and indeed, $supp(\exists_x c) \subseteq supp(c) \backslash \{x\}$.[6] Finally, the diagonal element $\delta_{x,y}$ has support $\{x, y\}$ for $x \neq y$, and $\delta_{x,x} = \bot$. Note also that these elements are not $\otimes$-compact.

Residuation works as expected (i.e., $(c_1 \oplus c_2)\eta = c_1\eta \oplus c_2\eta$), and top and bottom are the constant functions mapping all η to $\top$ and $\bot$, respectively.

Definition 16 (Soft CSPs). *A* soft constraint satisfaction problem *is a pair $\langle C, Y \rangle$, where $C \subseteq C$ is a set of constraints and $Y \subseteq V$ is a finite set of variables.*

[6] The operator is called *projection* in the soft framework, and $\exists_x c$ is denoted $c \Downarrow_{V-\{x\}}$.

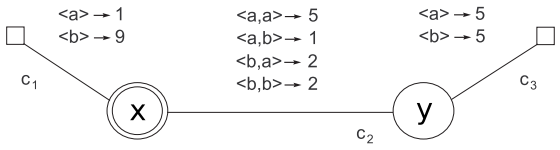

Figure 2: A soft CSP based on a Weighted CLIM.

The set Y contains the variables of interest for the constraint set C.

Definition 17 (Solutions). *The* solution *of a soft CSP $P = \langle C, Y \rangle$ is defined as the constraint $Sol(P) = \exists_{supp(\bigotimes C) \setminus Y} \bigotimes C$.*

The solution of a soft CSP is obtained by combining all constraints, and then hiding the variables that are not of interest. In this way we get the constraint with support (not greater than) Y "induced" by the entire CSP.

A tightly related notion, one which is quite important in combinatorial optimisation problems, is the best level of consistency [12].

Definition 18. *The* Best level of consistency *of a soft CSP problem $P = \langle C, Y \rangle$ is defined as the constraint $blevel(P) = \exists_Y Sol(P)$.*

Example 4. *Figure 2 shows a weighted CSP as a graph with values from the CLIM $\langle \mathbb{R}^+, \leq, + \rangle$ (the* Weighted *semiring, in the soft CSP jargon): the non-negative reals plus ∞ with the standard order and addition as monoidal operator. Variables and constraints are represented respectively by nodes and by undirected arcs (unary for c_1 and c_3, and binary for c_2), and values are written to the right of each tuple. The variables of interest are represented with a double circle (i.e. variable x). We assume that the domain of the variables contains only elements a and b. The solution of the weighted CSP of Figure 2 associates an element to every domain value of variable x. Such an element is obtained by first combining all the constraints together. For instance, for the tuple $\langle a, a \rangle$ (that is, $x = y = a$), we have to compute the sum of* 1 *(the value assigned to $x = a$ in constraint c_1),* 5 *(the value assigned to $\langle x = a, y = a \rangle$ in c_2) and* 5 *(the value assigned to $y = a$ in c_3). Hence, the resulting value for this tuple is* 11*. We can repeat the same procedure for tuple $\langle a, b \rangle \to 7$, $\langle b, a \rangle \to 16$ and $\langle b, b \rangle \to 16$. The variable y is then hidden in the resulting tuples, obtaining the solution $\langle a \rangle \to 7$ and $\langle b \rangle \to 16$. The* blevel *for the example in Figure 2 is* 7 *(related to the solution $x = a$, $y = b$).*

6. Examples of Computation

In the following we provide some examples of agent computations to clarify the semantics in Table **??** and Table **??**: we curb to the unlabelled versions without loss of generality, dropping the trailing **stop** whenever in parallel with another agent. We adopt the CLIM $\langle \mathbb{R}^+, \leq, + \rangle$, which considers non-negative reals plus ∞ with the standard order, and addition as the monoidal operator. We also consider two constraints $c_1 = x + 1$ ($sv(c_1) = \{x\}$) and $c_2 = y + 2$ ($sv(c_2) = \{y\}$).

Example 5 (Tell, ask, and parallel operators). *We start with the querying agent* $\mathbf{ask}(c_1)$*, executed in a store* $c_1 \otimes c_2$*: since* $c_1 \leq c_1 \otimes c_2$*,* $sv(c_1 \otimes c_2) = \{x, y\}$*, and* $sv(c_1) = \{x\}$*, using Table* **??** *we get* $\langle \mathbf{ask}(c_1) \to \mathbf{stop}, c_1 \otimes c_2 \rangle \mapsto_{\{x,y\}} \langle \mathbf{stop}, c_1 \otimes c_2 \rangle$*.*

Next example uses two agents in parallel each adding a different constraint. As stated in the precondition of rules **R1**-**R2** *in Table* **??***, the free variables of the suspended branch (i.e.,* $fv(B)$*) have to be in the subscript* Δ *as well, since* Δ *collects all the variables that are going to be used in all the branches. As an example, since* $fv(\mathbf{tell}(c_2)) = sv(c_2) = y$*, then* $\langle \mathbf{tell}(c_1) \parallel \mathbf{tell}(c_2), \bot \rangle \mapsto_{\{x,y\}} \langle \mathbf{tell}(c_2), c_1 \rangle \mapsto_{\{x,y\}} \langle \mathbf{stop}, c_1 \otimes c_2 \rangle$*.*

Example 6 (Procedure call and hiding). *Rule* **A3** *in Table* **??** *just requires the actual parameter* y *to be in the subscript* Δ *(together with the support variables of the global store, in this case* $sv(c_1) = \{x\}$*); then, it applies a straight renaming of the formal parameter with the actual one. An example is given below, where we suppose to have* $p(x) = \exists_x \mathbf{tell}(c_1)$ *among the procedures declared in* $\mathcal{P}$*, so that* $\langle p(y), c_1 \rangle \mapsto_{\{x,y\}} \langle \exists_y \mathbf{tell}(c_1[{}^y/_x]), c_1 \rangle \ldots$

Note that, by substituting x *with* y *in* c_1 *(i.e.,* $c_1[{}^y/_x]$*), the value becomes* $y + 1$ *(see Definition 11, and Proposition 3 for the operator instantiations): in words, the value of* x *in* c_1 *is passed to* y *with a diagonal constraint* ($\delta_{x,y} \otimes c_1$)*, and then the influence of* x *is removed from* c_1 *through the cylindric operator:* $\exists_x(\delta_{x,y} \otimes c_1)$*. The next step concerns the hiding operator: rule* **A4** *in Table* **??** *requires as precondition a fresh name* w ($w \notin \{x, y\}$) *that is used to rename the hidden variable* y*. In this case,* y *is renamed with* w *in* c_1 *(thus,* $d = w + 1$*, see Proposition 3), and* w *is then added to the subscript* Δ *because of the tell action* $\ldots \langle \exists_y \mathbf{tell}(c_1[{}^y/_x]), c_1 \rangle \mapsto_{\{x,y\}} \langle \mathbf{tell}((c_1[{}^y/_x])[{}^w/_y]), c_1 \rangle \mapsto_{\{x,y,w\}} \langle \mathbf{stop}, c_1 \otimes d \rangle$*.*

Therefore, the final store is $c_1 \otimes d = x + w + 2$*; notice that* y *has been recorded as used, but it is no longer present in the global store.*[7]

[7] Note $sv(c \otimes d) \subseteq sv(c) \cup sv(d)$, differently from what stated in [3], where = is assumed.

Fairness represents a large class of infinitary properties, usually viewed as a subclass of liveness properties (i.e., *"something good happens"*). Such properties restrict the set of potential computations of a system by disallowing those infinite computations that indefinitely delay some system component [21]. Fairness is known to affect liveness properties: the correctness of a program with respect to some property cannot be proved unless its infinite computations are restricted to just the fair ones. Unfortunately, transition rules defined in classical SCCP [3] do not consider infinite computations (crisp constraint-languages often treat them as failures [21]), something that we solve in this work instead. The next example illustrates how fair computations can be captured by the use of Δ with the semantics using a global store (Section **??**).

Example 7 (Fair computations). *In the following we consider an interaction $A \parallel B$ between a sequencer agent B that marks the rhythm of agent A, which in turn has the task to progressively consume an infinite resource by adding c_r each time. Constraints are indexed with their support variable, i.e., c_x, c_y, and c_r. A waits for B to grant permission to add c_r via a signal on c_x: when c_x is in the store, A adds c_r. When A finishes, it informs B by telling c_y, and B, hanging on the corresponding $\mathbf{ask}(c_y)$, is then unblocked. Finally, B launches the same parallel computation after changing the signalling variables x and y by renaming them (in the same way for both A and B). Two procedures $p_1(x, y)$ and $p_2(x, y)$ replicate the behaviour of the initial agents, so that they can be invoked infinite times.*

$$A = p_1(x, y) : \mathbf{ask}(c_x) \longrightarrow \mathbf{tell}(c_y \otimes c_r)$$

$$B = p_2(x, y) : \mathbf{tell}(c_x) \parallel (\mathbf{ask}(c_y) \longrightarrow \exists_{\{x,y\}}(p_1(x, y) \parallel p_2(x, y)))$$

Looking now at the reduction semantics provided in Table **??**, *the first observables are described from Eq. 9 to Eq. 15*

$$\langle A \parallel B, \bot\rangle \longrightarrow_{\{r,x,y\}} \langle A \parallel \mathbf{ask}(c_y) \longrightarrow \exists_{\{r,x,y\}}(p_1(x, y) \parallel p_2(x, y)), c_x\rangle \tag{9}$$

$$\begin{aligned}\langle \mathbf{tell}(c_y \otimes c_r) \parallel \mathbf{ask}(c_y) \longrightarrow \exists_{\{r,x,y\}}(p_1(x, y) \parallel p_2(x, y)), c_x\rangle \longrightarrow_{\{r,x,y\}} \\ \langle \mathbf{tell}(c_y \otimes c_r) \parallel \mathbf{ask}(c_y) \longrightarrow \exists_{\{r,x,y\}}(p_1(x, y) \parallel p_2(x, y)), c_x\rangle\end{aligned} \tag{10}$$

$$\begin{aligned}\langle \mathbf{tell}(c_y \otimes c_r) \parallel \mathbf{ask}(c_y) \longrightarrow \exists_{\{r,x,y\}}(p_1(x, y) \parallel p_2(x, y)), c_x\rangle \longrightarrow_{\{r,x,y\}} \\ \langle \mathbf{ask}(c_y) \longrightarrow \exists_{\{r,x,y\}}(p_1(x, y) \parallel p_2(x, y)), c_x \otimes c_y \otimes c_r\rangle\end{aligned} \tag{11}$$

$$\begin{aligned}\langle \mathbf{ask}(c_y) \longrightarrow \exists_{\{r,x,y\}}(p_1(x, y) \parallel p_2(x, y)), c_x\rangle \longrightarrow_{\{r,x,y\}} \\ \langle \exists_{\{r,x,y\}}(p_1(x, y) \parallel p_2(x, y)), c_x \otimes c_y \otimes c_r\rangle\end{aligned} \tag{12}$$

Then, if we rename x with v and y with w we have

$$\begin{aligned}\langle \exists_{\{r,x,y\}}(p_1(x,y) \parallel p_2(x,y)), c_x \otimes c_r \otimes c_y \otimes c_r\rangle \longrightarrow_{\{r,x,y\}} \\ \langle p_1(x,y)[^v/_x][^w/_y] \parallel p_2(x,y)[^v/_x][^w/_y], c_x \otimes c_y \otimes c_r\rangle\end{aligned} \tag{13}$$

where $p_1(x,y)[^v/_x][^w/_y] = p_1(v,w)$ *and* $p_2(x,y)[^v/_x][^w/_y] = p_2(v,w)$. *Then, if* $p_1(v,w)$ *is invoked first*

$$\begin{aligned}\langle p_1(v,w) \parallel p_2(v,w), c_x \otimes c_y \otimes c_r\rangle \longrightarrow_{\{r,v,w,x,y\}} \\ \langle \mathbf{ask}(c_v) \longrightarrow \mathbf{tell}(c_w \otimes c_r) \parallel p_2(v,w), c_x \otimes c_y \otimes c_r\rangle\end{aligned} \tag{14}$$

where $c_v = c_x[^v/_x]$ *and* $c_w = c_y \otimes c_r[^w/_y]$. *finally, also* $p_2(v,w)$ *is invoked*

$$\begin{aligned}\langle \mathbf{ask}(c_v) \longrightarrow \mathbf{tell}(c_w \otimes c_r) \parallel p_2(v,w), c_x \otimes c_y \otimes c_r\rangle \longrightarrow_{\{r,v,w,x,y\}} \\ \langle \mathbf{ask}(c_v) \longrightarrow \mathbf{tell}(c_w \otimes c_r) \parallel \mathbf{tell}(c_v) \parallel \\ \mathbf{ask}(c_w) \longrightarrow \exists_{\{v,w\}}(p_1(v,w) \parallel p_2(v,w)), c_x \otimes c_y \otimes c_r\rangle\end{aligned} \tag{15}$$

Then the computation continues as in Eq. 9, and so on. Note that the presented computation is fair because it satisfies Definition **??**.

7. Conclusions and Further Work

Inspired by the characterisation of the labelled semantics for the crisp variant of the language [2], in this paper we investigated observational and behavioural equivalences of the deterministic fragment of soft CCP [3]: we introduced the notion of $\otimes$-compactness, we proposed an observational semantics for the language and presented a novel behavioural characterisation in terms of weak bisimilarity (enhancing the syntactic bisimulation advanced in [6]). We then rounded the work by introducing an axiomatisation of the finite fragment of the language, in the spirit of [1].

The use of residuation theory (advanced in [14] for solving soft constraints problems) provides an elegant way to define the minimal information that enables the firing of actions in the LTS shown in Sec. 3. This choice allowed for the study of the behavioural equivalence of agents in terms of weak and strong bisimilarity on such LTS, and it allowed for relating them to the corresponding barbed bisimilarities of (unlabelled) reductions and with the standard semantics via fair computations. The two kinds of equivalences, as well as the axiomatisation for weak bisimilarity, are presented in this paper for the first time.

For future work, we plan to provide a denotational semantics for soft CCP by building on the work for the crisp case in [1]. No denotational semantics has been yet proposed for [3]. Concerning the axioms, we will try and investigate the relationship between soft CCP and a logical system whose fundamental properties are closely related to the ones we have investigated in this paper; namely *affine linear logic* [22]. This logical system *rejects contraction* but *admits weakening*, which intuitively correspond to dropping idempotence and preserving monotonicity in the soft formalism. The denotational model of CCP is based on *closure operators*: Each agent is compositionally interpreted as a monotonic, extensive and idempotent operator/function on constraints. We shall then investigate a denotational model for soft CCP processes based on *pre-closure operators* [23].

We plan to consider two extensions of the language, checking how far the results given in this paper can be adapted. As evidenced by [24], a non-deterministic extension is an interesting challenge since the closure under any context for the saturated bisimilarity gets more elaborated than just closing with respect to the addition of constraints (Defs. **??** and **??**, condition 3), and similarly one also needs to find the right formulation of bisimilarity for the labelled transitions systems. Also, the presence of residuation makes intuitive the definition of a retract operator for the calculus. Even if the operational semantics would be less affected, retraction would require a complete reformulation of the semantics via fair computations, since monotonicity (as stated in Lemma **??**) would not hold [5, 6].

Finally, we would like to find the same observational equivalences also for a semantics that considers local stores of agents. In this case, the hiding operator needs to carry some information on the variables it abstracts. According to [25], we should consider an extended operator $\exists_x^\sigma$, for σ the local store. The intuition is that variable x may be local to a component $\exists_x\sigma'$ of the store σ, yet visible at a global level: we must then evaluate A in the store when the local x is hidden, yet the possible duplications are removed (e.g., $\exists_x\sigma'$ may already occur in the global store σ). The final store contains in σ_1 the original σ' increased by what has been added by the step. To this end, we consider also to investigate the more general framework for distilling labelled semantics from unlabelled ones proposed in [26].

References

References

[1] V. A. Saraswat, M. C. Rinard, P. Panangaden, Semantic foundations of concurrent constraint programming, in: D. S. Wise (Ed.), POPL 1991, ACM Press, 1991, pp. 333–352.

[2] A. Aristizábal, F. Bonchi, C. Palamidessi, L. F. Pino, F. D. Valencia, Deriving labels and bisimilarity for concurrent constraint programming, in: M. Hofmann (Ed.), FOSSACS 2011, Vol. 6604 of LNCS, Springer, 2011, pp. 138–152.

[3] S. Bistarelli, U. Montanari, F. Rossi, Soft concurrent constraint programming, ACM Transactions on Computational Logic 7 (3) (2006) 563–589.

[4] M. G. Buscemi, U. Montanari, CC-Pi: A constraint-based language for specifying service level agreements, in: R. De Nicola (Ed.), ESOP 2007, Vol. 4421 of LNCS, Springer, 2007, pp. 18–32.

[5] S. Bistarelli, F. Santini, A secure non-monotonic soft concurrent constraint language, Fundamamenta Informaticae 134 (3-4) (2014) 261–285.

[6] S. Bistarelli, F. Santini, A secure non-monotonic soft concurrent constraint language, Fundam. Inform. 134 (3-4) (2014) 261–285.

[7] S. Bistarelli, M. Gabbrielli, M. C. Meo, F. Santini, Timed soft concurrent constraint programs, in: D. Lea, G. Zavattaro (Eds.), COORDINATION, Vol. 5052 of LNCS, 2008, pp. 50–66.

[8] S. Bistarelli, M. Gabbrielli, M. C. Meo, F. Santini, Timed soft concurrent constraint programs: An interleaved and a parallel approach, TPLP 15 (6) (2015) 743–782.

[9] F. Gadducci, F. Santini, L. F. Pino, F. D. Valencia, A labelled semantics for soft concurrent constraint programming, in: T. Holvoet, M. Viroli (Eds.), COORDINATION 2015, LNCS, Springer, 2015, pp. 133–149.

[10] G. Karner, Semiring-based constraint satisfaction and optimization, Semigroup Forum 45 (XX) (1992) 148–165.

[11] H. Fargier, J. Lang, Uncertainty in constraint satisfaction problems: a probabilistic approach, in: M. Clarke, R. Kruse, S. Moral (Eds.), ECSQUARU 1993, Vol. 747 of LNCS, Springer, 1993, pp. 97–104.

[12] S. Bistarelli, U. Montanari, F. Rossi, Semiring-based constraint satisfaction and optimization, Journal of ACM 44 (2) (1997) 201–236.

[13] J. Golan, Semirings and Affine Equations over Them: Theory and Applications, Kluwer, 2003.

[14] S. Bistarelli, F. Gadducci, Enhancing constraints manipulation in semiring-based formalisms, in: G. Brewka, S. Coradeschi, A. Perini, P. Traverso (Eds.), ECAI 2006, Vol. 141 of FAIA, IOS Press, 2006, pp. 63–67.

[15] N. Galatos, P. Jipsen, T. Kowalski, H. Ono, Residuated Lattices: An Algebraic Glimpse at Substructural Logics, Vol. 151, Elsevier, 2007.

[16] F. Baccelli, G. Cohen, G. Olsder, J.-P. Quadrat, Synchronization and Linearity: An Algebra for Discrete Event Systems, Wiley, 1992.

[17] J. D. Monk, An introduction to cylindric set algebras, Logic Journal of IGPL 8 (4) (2000) 451–496.

[18] U. Montanari, V. Sassone, Dynamic congruence vs. progressing bisimulation for CCS, Fundamenta informaticae 16 (2) (1992) 171–199.

[19] J. J. Leifer, R. Milner, Deriving bisimulation congruences for reactive systems, in: C. Palamidessi (Ed.), CONCUR 2000, LNCS, 2000, pp. 243–258.

[20] F. Bonchi, F. Gadducci, G. V. Monreale, Reactive systems, barbed semantics, and the mobile ambients, in: L. de Alfaro (Ed.), FOSSACS 2009, LNCS, 2009, pp. 272–287.

[21] M. Z. Kwiatkowska, Infinite behaviour and fairness in concurrent constraint programming, in: Sematics: Foundations and Applications, REX Workshop, Vol. 666 of Lecture Notes in Computer Science, Springer, 1992, pp. 348–383.

[22] U. Dal Lago, S. Martini, Phase semantics and decidability of elementary affine logic, Theoretical Computer Science 318 (3) (2004) 409–433.

[23] A.V.Arkhangelskii, L.S.Pontryagin, General Topology I, Springer, 1990.

[24] L. F. Pino, F. Bonchi, F. D. Valencia, A behavioral congruence for concurrent constraint programming with non-deterministic choice, in: G. Ciobanu, D. Méry (Eds.), ICTAC 2014, Vol. 8687 of LNCS, Springer, 2014, pp. 351–368.

[25] F. S. de Boer, M. Gabbrielli, E. Marchiori, C. Palamidessi, Proving concurrent constraint programs correct, ACM ToPLaS 19 (5) (1997) 685–725.

[26] F. Bonchi, F. Gadducci, G. V. Monreale, A general theory of barbs, contexts, and labels, ACM Transactions on Computational Logic 15 (4) (2014) 35:1–35:27.