Soft Concurrent Constraint Programming with Local Variables*

Laura Bussi[1], Fabio Gadducci[1], Francesco Santini[2]

[1] Dipartimento di Informatica, Università di Pisa, Pisa, Italy
laura.bussi@phd.unipi.it fabio.gadducci@unipi.it
[2] Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Perugia, Italy
francesco.santini@unipg.it

Abstract. We extend Soft Concurrent Constraint languages with the possibility to manage variables that are local (i.e., private) to some of the agents. Being constraints soft, it is possible to represent preferences as a partially ordered set. With respect to the related literature using an idempotent operator for constraint composition, a soft language requires a revision of the hiding operator, which is used to locally keep the computation effect on a variable, and conceal it from the global store. We provide the language with labelled and unlabelled reduction semantics as well as bisimulation equivalences, further proving their correspondence.

Keywords: Soft concurrent constraint programming, residuated monoids, local variables, bisimulation equivalences.

1 Introduction

Concurrent Constraint Programming (CCP) is a declarative model for concurrency where agents interact on a common store of information by telling and asking constraints [22]. In general terms, a constraint is a relationship on a set of variables: an assignment of (some of) the variables in the store needs to be found so to satisfy a given goal. A constraint system provides a signature from which the constraints are built; it is formalised as an algebra with operators to express conjunction of constraints, absent and inconsistent information, hiding of information and parameter passing.

The *polyadic* and *cylindric algebras* are two algebraisation of the first-order calculus [16], which have been widely adopted in the literature to provide the semantics of constraint formulas [17, 25]. A cylindric algebra is formed by enhancing a Boolean algebra by means of a family of unary operations called *cylindrifications*. Technically, the cylindrification operation $\exists_x(c)$ is used to project out the information about a variable x from a constraint c: for example, because it is important to focus only on the variables that appear in the goal of a constraint logic program.

While polyadic algebras are the algebraic version of the pure first-order calculus, cylindric algebras yield an algebraisation of the first-order calculus with *equality*. However, equality can be also achieved in polyadic algebras via additional axioms that specify which terms are to be considered equal under the abstract interpretation.

* Research partially supported by the MIUR PRIN 2017FTXR7S "IT-MaTTerS" and by GNCS-INdAM ("Gruppo Nazionale per il Calcolo Scientifico").

While most of the solutions in the literature adopt a cylindric algebra to represent constraints [25], other proposals take advantage of polyadic algebras: in [17] the motivation is to allow projections on infinite sets, while in [8] replacing diagonals (used to perform parameters passing [25], borrowed from cylindric algebras) with polyadic operators allows for a compact – *polynomial* – representation of soft constraints. Moreover, in case it is necessary to use preferences beside hard constraints, i.e. *Soft Concurrent Constraint Programming* (SCCP) [3, 15], algebra operators interact with a residuated monoid structure of values [14]: while the semi-lattice of such preferences must be complete for cylindric algebras, it is not necessary so for polyadic ones [8].

The soft CCP language we present in this paper is a further generalisation of what can be found in the literature, in particular [1] (see also Section 7). In particular, it allows for a more general algebraic structure than the absorptive and idempotent monoid used there, and it covers also bipolar (i.e., positive/negative) preferences, thus generalising [6]; secondly, polyadic algebras can model many problems using a polynomial representation of constraints. In fact, polynomial constraints play an important role in program analysis and verification (e.g. when synthesising program invariants and analysing the reachability of hybrid systems), and they have been recently used in SAT modulo theories [9]. Moreover, the language allows an agent to perform operations on variables (in particular, adding and asking constraints) that are local, i.e., visible only to the agent itself: for this reason, the *hiding* operator needs to consider the effect of local steps with respect to the global store, which is seen by all the agents participating to a concurrent computation. In this way, it is possible to distinguish between local and global knowledge of agents, in the form of a local and a global store of constraints.

Beside the language syntax, we provide a reduction semantics and a saturated bisimulation relation, again taking inspiration from and generalising [1]. In order for two computation states to be *saturated bisimilar*, it is required that *i)* they should expose the same barbs, *ii)* whenever one of them moves then the other should reply and arrive at an equivalent state, *iii)* they should be equivalent under all the possible stores. Intuitively, barbs are basic observations (i.e., predicates) on the system states; in the case of CCP languages, barbs are represented by the constraint store. In addition, we show a labelled bisimulation to (partially) overcome the need to check the store closure (i.e., item *iii* in the previous paragraph). As a final step, we show that the labelled and unlabelled reduction semantics correspond, and we advance a labelled bisimulation relation.

This paper is a continuation of [8], exploiting the polyadic formalism to define a concurrent constraint language. Section 2 and Section 3 present the necessary background on the algebraic structure needed to model polyadic constraints. The following sections are focused on the semantics of a concurrent constraint-based language using local variables and polyadic constraints, on the correspondence between different semantics, and on the equivalence relations among processes. Section 4 presents the syntax and a reduction semantics for the language, while Section 5 presents a labelled reduction for the same language. Section 6 shows further formal results on the correspondence between the two semantics, and a bisimilarity relation to compare processes with the labelled semantics. In Section 7 we summarise the most related work about CCP-based languages with the notion of local and global variables. In Section 8 we finally wrap up the paper with conclusive thoughts and ideas about future works.

2 An introduction to Residuated Monoids

This section reports some results on residuated monoids, which are the algebraic structure adopted for modelling soft constraints in the following of the paper. These background results are mostly drawn from [15], where also proofs can be found.

2.1 Preliminaries on Ordered Monoids

The first step is to define an algebraic structure for modelling preferences, where it is possible to compare values and combine them. Our choice falls into the range of *bipolar* approaches, in order to represent both positive and negative preferences: we refer to [14] for a detailed introduction and a comparison with other proposals.

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

The LUB of a (possibly infinite) subset $X \subseteq A$ is denoted $\bigvee X$, and it is clearly unique. Note that $\bigvee \emptyset$ is the bottom, denoted as $\bot$, of the PO. Should it exist, $\bigvee A$ is the top, denoted as $\top$, of the PO.

Definition 2 (Ordered monoids). *A (commutative) monoid is a triple $\langle A, \otimes, \mathbf{1} \rangle$ such that $\otimes : A \times A \to A$ is a commutative and associative function and $\mathbf{1} \in A$ is its* identity *element, i.e., $\forall a \in A. a \otimes \mathbf{1} = a$. A partially ordered monoid (POM) is a 4-tuple $\langle A, \leq, \otimes, \mathbf{1} \rangle$ such that $\langle A, \leq \rangle$ is a PO and $\langle A, \otimes, \mathbf{1} \rangle$ a monoid. A semi-lattice monoid (SLM) is a POM such that their underlying PO is a SL.*

As usual, we use the infix notation: $a \otimes b$ stands for $\otimes(a, b)$.

Example 1 (Power set). Given a (possibly infinite) set S, a key example is represented by the POM $\mathbb{P}(S) = \langle 2^S, \subseteq, \cap, S \rangle$ of subsets of S, with the partial order given by subset inclusion and the (idempotent) monoidal operator by intersection. In fact, $\mathbb{P}(S)$ is a continuous lattice, since all LUBs exist, and S is both the top and the identity element.

In general, the partial order $\leq$ and the multiplication operator $\otimes$ can be unrelated. This is not the case for distributive SLMs (such as $\mathbb{P}(S)$ above).

Definition 3 (Distributivity). *Let $\langle A, \leq, \otimes, \mathbf{1} \rangle$ be an SLM. It is distributive if for any finite $X \subseteq A$ it holds $\forall a \in A. a \otimes \bigvee X = \bigvee \{a \otimes x \mid x \in X\}$.*

Note that distributivity implies that $\otimes$ is monotone with respect to $\leq$.

Remark 1. It is almost straightforward to show that our proposal encompasses many other formalisms in the literature. Indeed, distributive semi-lattice monoids are *tropical* semirings (also known as dioids), namely, semirings with an idempotent sum operator $a \oplus b$, which in our formalism is obtained as $\bigvee \{a, b\}$. If $\mathbf{1}$ is the top of the SL we end up in *absorptive* semirings [19], which are known as *c*-semirings in the soft constraint jargon [3] (see [4] for a brief survey on residuation for such semirings). Note that requiring the monotonicity of $\otimes$ and imposing $\mathbf{1}$ to be the top of the partial order means that preferences are negative, i.e., that it holds $\forall a, b \in A. a \otimes b \leq a$.

2.2 Remarks on residuation

It is often needed to be able to "remove" part of a preference, due e.g. to the non-monotone nature of the language at hand for manipulating constraints. The structure of our choice is given by residuated monoids [19]. They introduce a new operator $\oslash$, which represents a "weak" (due to the presence of partial orders) inverse of $\otimes$.

Definition 4 (Residuation). *A residuated POM is a 5-tuple $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ such that $\langle A, \leq, \otimes, \mathbf{1} \rangle$ is a partially ordered monoid and $\oslash : A \times A \to A$ is a function satisfying $\forall a, b, c \in A. b \otimes c \leq a \iff c \leq a \oslash b$. A residuated SLM is a residuated POM such that the underlying PO is a SL.*

In order to confirm the intuition about weak inverses, Lemma 1 below precisely states that residuation conveys the meaning of an approximated form of subtraction.

Lemma 1. *Let $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ be a residuated POM. Then $a \oslash b = \bigvee \{c \mid b \otimes c \leq a\}$ for all $a, b \in A$.*

In words, the LUB of the (possibly infinite) set $\{c \mid b \otimes c \leq a\}$ exists and is equal to $a \oslash b$. In fact, residuation implies distributivity (see [14, Lemma 2.2]).

Lemma 2. *Let $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ be a residuated POM. Then $\otimes$ is monotone. If additionally it is a SLM, then it is distributive.*

Example 2. Consider again the SLM $\mathbb{P}(S)$ from Example 1. It is clearly residuated, with $X \oslash Y = (S \setminus Y) \cup X$. In fact, the residuated operator plays the role of classical logical implication $Y \implies X$. Note also that $S \setminus Y = \emptyset \oslash Y$, so algebraically we have that $X \oslash Y = X \vee (\bot \oslash Y)$ holds in $\mathbb{P}(S)$.

In any residuated POM the $\oslash$ operator is also monotone on the first argument and anti-monotone on the second one, i.e., $\forall a, b, c \in A. a \leq b \implies c \oslash b \leq c \oslash a$. The definition below identifies sub-classes of residuated monoids that are suitable for an easier manipulation of constraints (see e.g. [4]).

Definition 5 (Families of POMs). *A residuated POM $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ is*

- localised *if* $\forall a \in A. a \notin \{\bot, \top\} \implies a \oslash a = \mathbf{1}$*;*
- invertible *if* $\forall a, b \in A. a \leq b < \top \implies b \otimes (a \oslash b) = a$*;*
- cancellative *if* $\forall a, b, c \in A. a \notin \{\bot, \top\} \wedge a \otimes b = a \otimes c \implies b = c$.

Remark 2. When introduced in [14, Def. 2.4], localisation was equivalently stated as $\forall a, b \in A. \bot < a \leq b < \top \implies a \oslash b \leq \mathbf{1}$. Indeed, the latter implies $a \oslash a \leq \mathbf{1}$, while $\mathbf{1} \leq a \oslash a$ by definition. Now, assuming $a \oslash a = \mathbf{1}$ and $a \leq b$, by anti-monotonicity $a \oslash b \leq a \oslash a = \mathbf{1}$. Note the constraint on $a \notin \{\bot, \top\}$: indeed, a residuated POM always has a top element and moreover $a \oslash \bot = \top \oslash a = \top$ for any a.

Note that being cancellative is a strong requirement. It implies e.g. some uniqueness of invertibility, that is, for any a, b there exists at most a c such that $b \otimes c = a$. It is moreover equivalent to what we could call strong locality, that is, $\forall a, b \in A. a \notin \{\bot, \top\} \implies (a \otimes b) \oslash a = b$. Indeed, this property implies cancellativeness, since if $a \otimes b = a \otimes c$ then $b = (a \otimes b) \oslash a = (a \otimes c) \oslash a = c$. On the other side, it is implied, since $((a \otimes b) \oslash a) \otimes a = a \otimes b$ holds in residuated POMs.

As a final remark, note that $\mathbb{P}(S)$ is localised and invertible, yet it is not cancellative.

3 A polyadic approach to constraint manipulation

This section presents our personal take on polyadic algebras for ordered monoids: the standard axiomatisation of e.g. [24] has been completely reworked, in order to be adapted to the constraints formalism. It extends our previous description in [8] by further elaborating on the laws for the polyadic operators in residuated monoids.

3.1 Cylindric and Polyadic operators for Ordered Monoids

We introduce two families of operators that will be used for modelling variables hiding and substitution, which are key features in languages for manipulating constraints. One is a well-known abstraction for existential quantifiers, the other one an axiomatisation of the notion of substitution, and it is proposed as a weaker alternative to diagonals [25], the standard tool for modelling equivalence in constraint programming.[3]

Cylindric operators. We fix a POM $\mathbb{S} = \langle A, \leq, \otimes, \mathbf{1} \rangle$ and a set V of variables, and we define a family of cylindric operators axiomatising existential quantifiers.

Definition 6 (Cylindric ops). *A cylindric operator $\exists$ for $\mathbb{S}$ and V is a family of monotone 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. $a \leq \exists_x a$,
2. $\exists_x \exists_y a = \exists_y \exists_x a$,
3. $\exists_x (a \otimes \exists_x b) = \exists_x a \otimes \exists_x b$.

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

In other words, $\exists$ fixes a monoid action which is monotone and increasing.

Polyadic operators. We now move to define a family of operators axiomatising substitutions. They interact with quantifiers, thus, beside a partially ordered monoid $\mathbb{S}$ and a set V of variables, we fix a cylindric operator $\exists$ over $\mathbb{S}$ and V.

Let $F(V)$ be the set of functions with domain and codomain V. For a function σ its support is $sv(\sigma) = \{x \mid \sigma(x) \neq x\}$ and, for a set $X \subseteq V$, $\sigma \mid_X : X \to V$ denotes the restriction of σ to X and $\sigma^c(X) = \{y \mid \sigma(y) \in X\}$ the counter-image of X along σ.

Definition 7 (Polyadic ops). *A polyadic operator s for a cylindric operator $\exists$ is a family of monotone functions $s_\sigma : A \to A$ indexed by elements in $F(V)$ such that for all $a, b \in A$, $x \in V$, and $\sigma, \tau \in F(V)$*

1. $sv(\sigma) \cap sv(a) = \emptyset \implies s_\sigma a = a$,
2. $s_\sigma(a \otimes b) = s_\sigma a \otimes s_\sigma b$,
3. $\sigma \mid_{sv(a)} = \tau \mid_{sv(a)} \implies s_\sigma a = s_\tau a$,
4. $\exists_x s_\sigma a = \begin{cases} s_\sigma \exists_y a & \text{if } \sigma^c(x) = \{y\} \\ s_\sigma a & \text{if } \sigma^c(x) = \emptyset \end{cases}$.

A polyadic operator offers enough structure for modelling variable substitution. In the following, we fix a cylindric operator $\exists$ and a polyadic operator s for it.

[3] "Weaker alternative" here means that diagonals allow for axiomatising substitutions at the expenses of working with complete partial orders: see e.g. [15, Definition 11].

3.2 Cylindric and Polyadic operators for Residuated Monoids

We now consider the interaction of previous structures with residuation. To this end, in the following we assume that $\mathbb{S}$ is a residuated POM (see Def. 4).

Lemma 3. *Let $x \in V$ and $a, b \in A$. Then it holds $\exists_x(a \oplus \exists_x b) \leq \exists_x a \oplus \exists_x b \leq \exists_x(\exists_x a \oplus b)$.*

Remark 3. It is easy to check that $\exists_x(a \oplus \exists_x b) \leq \exists_x a \oplus \exists_x b$ is actually equivalent to state that $\exists_x(a \otimes \exists_x b) \geq \exists_x a \otimes \exists_x b$.

We can show that $\oplus$ does not substantially alter the free variables of its arguments.

Lemma 4. *Let $a, b \in A$. Then it holds $sv(a \oplus b) \subseteq sv(a) \cup sv(b)$.*

A result similar to Lemma 3 relates residuation and polyadic operators.

Lemma 5. *Let $a, b \in A$ and $\sigma \in F(V)$. Then it holds $s_\sigma(a \oplus b) \leq s_\sigma a \oplus s_\sigma b$. Furthermore, if σ is invertible, then the equality holds.*

3.3 Polyadic Soft Constraints

Our key example comes from the soft constraints literature: our presentation generalises [5], whose underlying algebraic structure is of absorptive and idempotent semirings.

Definition 8 (Soft constraints). *Let V be a set of variables, D a finite domain of interpretation and $\mathbb{S} = \langle A, \leq, \otimes, \oplus, \mathbf{1} \rangle$ a residuated SLM. A* (soft) constraint *$c : (V \to D) \to A$ is a function associating a value in A with each assignment $\eta : V \to D$ of the variables.*

The set of constraints forms a residuated SLM $\mathbb{C}$, with the structure lifted from $\mathbb{S}$. Denoting the application of a constraint function $c : (V \to D) \to A$ to a variable assignment $\eta : V \to D$ as $c\eta$, we have that $c_1 \leq c_2$ if $c_1\eta \leq c_2\eta$ for all $\eta : V \to D$.

Lemma 6 (Cylindric and polyadic operators for (soft) constraints). *The residuated SLM of constraints $\mathbb{C}$ admits cylindric and polyadic operators, defined as*

- $(\exists_x c)\eta = \bigvee\{c\rho \mid \eta \mid_{V\setminus\{x\}} = \rho \mid_{V\setminus\{x\}}\}$ *for all $x \in V$,*
- $(s_\sigma c)\eta = c(\eta \circ \sigma)$ *for all $\sigma \in F(V)$.*

Remark 4. Note that $sv(c)$ coincides with the classical notion of support for soft constraints. Indeed, if $x \notin sv(c)$, then two assignments $\eta_1, \eta_2 : V \to D$ differing only for the image of x coincide (i.e., $c\eta_1 = c\eta_2$). The cylindric operator is called *projection* in the soft framework, and $\exists_x c$ is denoted $c \Downarrow_{V\setminus\{x\}}$.

Example 3. For the sake of simplicity, and to better illustrate the differences of our proposal with respect to [1], our running example will be the SLM of soft constraints where D is a finite initial segment of the naturals and $\mathbb{S}$ is the residuated SLM $\langle\{\bot, \top\}, \{\bot \leq \top\}, \wedge, \top\rangle$ of booleans. That SLM coincide with the SLM $\mathbb{P}(F)$, for F the family of functions $V \to D$: it is localised and invertible, and the top and the identity element coincides with F, i.e., the constraint c such that $c\eta = \top$ for all η. We will then usually express a constraint in $\mathbb{P}(F)$ as a SAT formula with inequations like $x \leq 1$, intended as $(x \leq 1)\eta = \top$ if $\eta(x) \leq 1$ and $\bot$ otherwise. The support of $x \leq 1$ is of course $\{x\}$. As expected, $\exists_X$ behaves as an existential quantifier, so that $\exists_x(x \leq 1) = \top$ and $\exists_x((x \leq 1) \wedge (y \leq 3)) = y \leq 3$. Similarly, for a substitution σ we have that $s_\sigma(x \leq 1) = \sigma(x) \leq 1$.

4 Polyadic Soft CCP: Syntax and reduction semantics

This section introduces our language. We fix a set of variables V, ranged over by x, y, ..., and a residuated POM $\mathbb{S} = \langle \mathscr{C}, \leq, \otimes, \oplus, \mathbf{1} \rangle$, which is cylindric and polyadic over V and whose elements are ranged over by c, d, ...

Definition 9 (Agents). *The set $\mathscr{A}$ of agents, parametric with respect to a set $\mathscr{P}$ of (unary) procedure declarations $p(x) = A$, is given by the following grammar*

$$A ::= \mathbf{stop} \mid \mathbf{tell}(c) \mid \mathbf{ask}(c) \mapsto A \mid A \parallel A \mid p(x) \mid \exists_x A$$

Hence, the syntax includes a termination agent **stop**, and the two typical operations of CCP languages [25]: **tell**(c) adds the constraint c to a common store through which all agents interact, and $\mathbf{ask}(c) \mapsto A$ continues as agent A only when c is entailed by such a store (otherwise its execution is suspended). The other operators respectively express the parallel composition between two agents (i.e., $A \parallel A$), the hiding of a variable x in the computation of A ($\exists_x A$), and, finally, the calling of a procedure $p \in \mathscr{P}$ (whose body is an agent A) with an actual parameter identified by variable x.

In the following we consider a set $\mathscr{E}$ of extended agents that uses the existential operator $\exists_x^{\pi} A$, where $\pi \in \mathscr{C}^*$ is meant to represent the sequence of updates performed on the local store. More precisely, the extended agent may carry some information about the hidden variable x in an incremental way. We will often write $\exists_x A$ for $\exists_x^{[]} A$ and π_i for the i-th element of $\pi = [\pi_0, \ldots, \pi_n]$.

We denote by $fv(A)$ the set of free variables of an (extended) agent, defined in the expected way by structural induction, assuming that $fv(\mathbf{tell}(c)) = sv(c)$, $fv(\mathbf{ask}(c) \mapsto A) = sv(c) \cup fv(A)$, and $fv(\exists_x^{\pi} A) = (fv(A) \cup \bigcup_i sv(\pi_i)) \setminus \{x\}$. In the following, we restrict our attention to procedure declarations $p(x) = A$ such that $fv(A) = \{x\}$.

Definition 10 (Substitutions). *Let $[^y/_x] : V \to V$ be the substitution defined as*

$$[^y/_x](w) = \begin{cases} y & \text{if } w = x \\ w & \text{otherwise} \end{cases}.$$

It induces an operator $[^y/_x] : \mathscr{E} \to \mathscr{E}$ on extended agents as expected, in particular

$$(\exists_w^{\pi} A)[^y/_x] = \begin{cases} \exists_w^{(s_{[^y/_x]}\pi)} A[^y/_x] & \text{if } w \notin \{x, y\} \\ (\exists_z^{(s_{[^z/_w]}\pi)} A[^z/_w])[^y/_x] & \text{for } z \notin fv(\exists_w^{\pi} A) \text{ otherwise} \end{cases}$$

where $s_{[^y/_x]} : \mathscr{C} \to \mathscr{C}$ is the function associated with $[^y/_x]$ and $s_{[^y/_x]}[\pi_1, \ldots, \pi_n]$ is a shorthand for $[s_{[^y/_x]}\pi_1, \ldots, s_{[^y/_x]}\pi_n]$.

Note that the choice of z in the rule above is immaterial, since for the polyadic operator it holds $\exists_x c = \exists_y s_{[^y/_x]}(c)$ if $y \notin sv(c)$. In the following we consider terms to be equivalent up-to α-conversion, meaning that terms differing only for hidden variables are considered equivalent, i.e., $\exists_w^{\pi} A = \exists_z^{(s_{[^z/_w]}\pi)} A[^z/_w]$ for $z \notin fv(\exists_w^{\pi} A)$.

Lemma 7. *Let $A \in \mathscr{E}$ and $x \notin fv(A)$. Then $A[^y/_x] = A$.*

$$\textbf{A1}\ \langle \textbf{tell}(c), \sigma \rangle \to \langle \textbf{stop}, \sigma \otimes c \rangle \quad \textbf{Tell}$$

$$\textbf{A2}\ \frac{\sigma \leq c}{\langle \textbf{ask}(c) \mapsto A, \sigma \rangle \to \langle A, \sigma \rangle} \quad \textbf{Ask}$$

$$\textbf{A3}\ \frac{p(x) = A \in \mathscr{P}}{\langle p(y), \sigma \rangle \to \langle A[^{y}/_{x}], \sigma \rangle} \quad \textbf{Rec}$$

Table 1. Axioms of the reduction semantics for SCCP.

Example 4. Consider the SLM $\mathbb{P}(F)$ illustrated in Example 3. We can specify agents such as $\textbf{ask}(y \leq 5) \mapsto \textbf{stop}$, i.e., an agent asking the store about the possible values of y, then terminating. Or $\exists_x(\textbf{tell}(x \leq 1) \parallel \textbf{tell}(y \leq 3))$ with $x \neq y$, meaning that the constraint $x \leq 1$ is local: indeed, thanks to α-conversion it coincides with $\exists_z(\textbf{tell}(z \leq 1) \parallel \textbf{tell}(y \leq 3))$ for any $z \neq y$. As we are going to see, the execution of $\textbf{tell}(z \leq 1)$ will take the latter agent to $\exists_z^{[z \leq 1]}\textbf{tell}(y \leq 3)$, which in turn coincides with $\exists_x^{[x \leq 1]}\textbf{tell}(y \leq 3)$.

4.1 Reduction semantics

We now move to the reduction semantics of our calculus. Given a sequence $\pi = [\pi_1, \ldots, \pi_n]$, we will use $\pi_\otimes$ and $\exists_x \pi$ as shorthands for $\pi_1 \otimes \ldots \otimes \pi_n$ and $[\exists_x \pi_1, \ldots, \exists_x \pi_n]$, respectively, sometimes combining them as in $(\exists_x \pi)_\otimes$, with $[]_\otimes = \mathbf{1}$.

Definition 11 (Reductions). *Let $\Gamma = \mathscr{E} \times \mathscr{C}$ be the set of* configurations. *The* direct reduction semantics *for SCCP is the pair $\langle \Gamma, \to \rangle$ such that $\to \subseteq \Gamma \times \Gamma$ is the binary relations obtained by the axioms in Table 1.*

The reduction semantics *for SCCP is the pair $\langle \Gamma, \to \rangle$ such that $\to \subseteq \Gamma \times \Gamma$ is the binary relation obtained by the rules in Table 1 and Table 2.*

$$\textbf{R1}\quad \frac{\langle A, \sigma \rangle \to \langle A', \sigma' \rangle}{\langle A \parallel B, \sigma \rangle \to \langle A' \parallel B, \sigma' \rangle} \quad \textbf{Par1}$$

$$\textbf{R2}\ \frac{\langle A, \pi_0 \otimes \sigma \rangle \to \langle B, \sigma_1 \rangle \text{ with } \pi_0 = \pi_\otimes \oplus (\exists_x \pi)_\otimes}{\langle \exists_x^{\pi} A, \sigma \rangle \to \langle \exists_x^{\pi\rho} B, \sigma \otimes \exists_x \rho \rangle \text{ with } \rho = \sigma_1 \oplus (\pi_0 \otimes \sigma)} \text{ for } x \notin sv(\sigma)\ \textbf{Hide}$$

Table 2. Contextual rules of the reduction semantics for SCCP.

The split distinguishes between the axioms and the rules guaranteeing the closure with respect to the parallel and existential operators. Indeed, rule **R1** models the interleaving of two agents in parallel, assuming for the sake of simplicity that the parallel operator is associative and commutative, as well as satisfying $\textbf{stop} \parallel A = A$. In **A1** a constraint c is added to the store σ. **A2** checks if c is entailed by σ: if not, the computation is blocked. Axiom **A3** replaces a procedure identifier with the associated body, renaming the formal parameter with the actual one.

Let us instead discuss in some details the rule **R2**. The intuition is that if we reach an agent $\langle \exists_x^{\pi} A, \sigma \rangle$, then during the computation a sequence π of updates has been performed by the local agent, and $(\exists_x \pi)_{\otimes}$ has been added to the global store. The chosen store for the configuration in the premise is $\pi_0 \otimes \sigma$ for $\pi_0 = \pi_{\otimes} \oplus (\exists_x \pi)_{\otimes}$: the effect $(\exists_x \pi)_{\otimes}$ of the sequence of updates is removed from the local store $\pi_{\otimes}$, which may carry information about x, since that effect had been previously added to the global store. Now, $\rho = \sigma_1 \oplus (\pi_0 \otimes \sigma)$ is precisely the information added by the step originating from A, which is then restricted and added to σ. On the local store we simply add that effect ρ to the sequence of updates, with $\pi\rho = [\pi_0, \ldots, \pi_n, \rho]$.

Lemma 8 (On monotonicity). *Let $\langle A, \sigma \rangle \to \langle B, \rho \rangle$ be a reduction. Then $\rho = (\rho \oplus \sigma) \otimes \sigma$ and $fv(\langle B, \rho \rangle) \subseteq fv(\langle A, \sigma \rangle)$.*

Example 5. Consider the agents $A_1 = \mathbf{ask}(y \leq 5) \mapsto \mathbf{stop}$ and $A_2 = \exists_x(\mathbf{tell}(x \leq 1) \parallel \mathbf{tell}(y \leq 3))$ with $x \neq y$ discussed in Example 4, and the configuration $\langle A_1 \parallel A_2, \top \rangle$. Starting from the configuration $\langle A_2, \top \rangle$ we have the reductions

$$\langle A_2, \top \rangle \to \langle \exists_x^{[y \leq 3]} \mathbf{tell}(x \leq 1), y \leq 3 \rangle \to \langle \exists_x^{[y \leq 3, y > 3 \vee x \leq 1]} \mathbf{stop}, y \leq 3 \rangle$$

In both cases, first we apply **A1**, then **R1** and finally **R2**. Looking at the application of **R2** to the first reduction, we have that $\pi_{\otimes} = \top = (\exists_x \pi)_{\otimes}$, thus $\pi_0 = \top$, $\rho = (y \leq 3) \oplus \top = y \leq 3 = \exists_x (y \leq 3)$. Now, consider the second reduction. In that case we have $\pi_0 = (y \leq 3) \oplus (y \leq 3) = \top$ and $\rho = ((y \leq 3) \wedge (x \leq 1)) \oplus (y \leq 3) = y > 3 \vee x \leq 1$ and $\exists_x \rho = \top$. Note that in both the second and third state, agent A_1 could be executed.

Remark 5. With respect to the crisp language with local variables introduced in [1], which can be recast in our framework as absorptive POMs where the monoidal operator is idempotent, our proposal differs mostly for the structure of rule **R2**, which could be presented as shown below

$$\frac{\langle A, \pi_0 \otimes \sigma \rangle \to \langle B, \xi \otimes \pi_0 \otimes \sigma \rangle \text{ with } \pi_0 = \pi_{\otimes} \oplus (\exists_x \pi)_{\otimes}}{\langle \exists_x^{\pi} A, \sigma \rangle \to \langle \exists_x^{\pi \xi} B, \sigma \otimes \exists_x \xi \rangle} \text{ for } x \notin sv(\sigma)$$

The proposals coincide for cancellative monoids, since inverses are unique. However, this is not so if the monoidal operator is idempotent, thus the crisp rule represents in fact a schema, giving rise to a possibly infinite family of reductions departing from an agent. Our choice of the witness $\exists_x \sigma_1 \oplus (\pi_0 \otimes \sigma)$ avoids such non-determinism.

Let $\gamma = \langle A, \sigma \rangle$ be a configuration. We denote by $fv(\gamma)$ the set $fv(A) \cup sv(\sigma)$ and by $\gamma[^z/_w]$ the component-wise application of the substitution $[^z/_w]$.

Definition 12. *A configuration $\langle A, \sigma \rangle$ is initial if $A \in \mathscr{A}$ and $\sigma = \mathbf{1}$; it is reachable if it can be reached by an initial configuration via a sequence of reductions.*

Lemma 9 (On monotonicity, II). *Let $\langle A \parallel \exists_x^{\pi} B, \sigma \rangle$ be a reachable configuration. Then $\sigma = (\sigma \oplus (\exists_x \pi)_{\otimes}) \otimes (\exists_x \pi)_{\otimes}$.*

Remark 6. An alternative solution for the the structure of rule **R2** would have been

$$\frac{\langle A, \pi_\otimes \otimes \sigma_0\rangle \to \langle B, \sigma_1\rangle \text{ with } \sigma_0 = \sigma \oplus (\exists_x \pi)_\otimes}{\langle \exists_x^\pi A, \sigma\rangle \to \langle \exists_x^{\pi\rho} B, \sigma \otimes \exists_x \rho\rangle \text{ with } \rho = \sigma_1 \oplus (\pi_\otimes \otimes \sigma_0)} \text{ for } x \notin sv(\sigma)$$

Indeed, in the light of Lemma 9, the proposals coincide for invertible semirings, since $\pi_0 \otimes \sigma = (\pi_\otimes \oplus (\exists_x \pi)_\otimes) \otimes (\exists_x \pi)_\otimes \otimes (\sigma \oplus (\exists_x \pi)_\otimes) \leq \pi_\otimes \otimes (\sigma \oplus (\exists_x \pi)_\otimes)$, and the equality holds for invertible semirings since $\pi_\otimes \leq (\exists_x \pi)_\otimes$.

4.2 Saturated bisimulation

As proposed in [1] for crisp languages, we define a barbed equivalence between two agents [21]. Intuitively, barbs are basic observations (predicates) on the states of a system, and in our case they correspond to the constraints in $\mathscr{C}$.

Definition 13 (Barbs). *Let $\langle A, \sigma\rangle$ be a configuration and $c \in \mathscr{C}$. We say that $\langle A, \sigma\rangle$ verifies c, or that $\langle A, \sigma\rangle \downarrow_c$ holds, if $\sigma \leq c$.*

Satisfying a barb c means that the agent $\mathbf{ask}(c) \mapsto A$ can be executed in the store σ, i.e., the reduction $\langle \mathbf{ask}(c) \mapsto A, \sigma\rangle \to \langle A, \sigma\rangle$ is allowed. We now move to equivalences: along [1], we propose the use of *saturated bisimilarity* to obtain a congruence.

Definition 14 (Saturated bisimilarity). *A saturated bisimulation is a symmetric relation R on configurations such that whenever $(\langle A, \sigma\rangle, \langle B, \rho\rangle) \in R$*

1. *if $\langle A, \sigma\rangle \downarrow_c$ then $\langle B, \rho\rangle \downarrow_c$;*
2. *if $\langle A, \sigma\rangle \to \gamma_1$ then there is γ_2 such that $\langle B, \rho\rangle \to \gamma_2$ and $(\gamma_1, \gamma_2) \in R$;*
3. *$(\langle A, \sigma \otimes d\rangle, \langle B, \rho \otimes d\rangle) \in R$ for all d.*

We say that γ_1 and γ_2 are saturated bisimilar ($\gamma_1 \sim_s \gamma_2$) if there exists a saturated bisimulation R such that $(\gamma_1, \gamma_2) \in R$. We write $A \sim_s B$ if $\langle A, \mathbf{1}\rangle \sim_s \langle B, \mathbf{1}\rangle$.

Note that $\langle A, \sigma\rangle \sim_s \langle B, \rho\rangle$ implies that $\sigma = \rho$. Moreover, it is also a congruence. Indeed, a context $C[\cdot]$, i.e., an agent with a placeholder $\cdot$, can modify the behaviour of a configuration only by adding constraints to its store.

Proposition 1. *Let $A \sim_s B$ and $C[\cdot]$ a context. Then $C[A] \sim_s C[B]$.*

5 Labelled reduction semantics

The definition of $\sim_s$ is unsatisfactory because of the store closure, i.e., the quantification in condition *3* of Definition 14. This section presents a labelled version of the reduction semantics that allows for partially avoiding such drawback.

Definition 15 (Labelled reductions). *Let $\Gamma = \mathscr{A} \times \mathscr{C}$ be the set of* configurations. *The* labelled direct reduction semantics *for SCCP is the pair $\langle \Gamma, \to\rangle$ such that $\to \subseteq \Gamma \times \mathscr{C} \times \Gamma$ is the ternary relation obtained by the axioms in Table 3.*

The labelled reduction semantics *for SCCP is the pair $\langle \Gamma, \to\rangle$ such that $\to \subseteq \Gamma \times \mathscr{C} \times \Gamma$ is the ternary relation obtained by the rules in Table 3 and Table 4.*

$$\textbf{LA1}\quad \langle \textbf{tell}(c), \sigma\rangle \xrightarrow{\mathbf{1}} \langle \textbf{stop}, \sigma \otimes c\rangle \qquad \textbf{Tell}$$

$$\textbf{LA2}\quad \frac{\alpha \leq c \oplus \sigma}{\langle \textbf{ask}(c) \mapsto A, \sigma\rangle \xrightarrow{\alpha} \langle A, \alpha \otimes \sigma\rangle} \qquad \textbf{Ask}$$

$$\textbf{LA3}\quad \frac{p(x) = A \in \mathscr{P}}{\langle p(y), \sigma\rangle \xrightarrow{\mathbf{1}} \langle A[{}^{y}/_{x}], \sigma\rangle} \qquad \textbf{Rec}$$

Table 3. Axioms of the labelled semantics for SCCP.

$$\textbf{LR1}\quad \frac{\langle A, \sigma\rangle \xrightarrow{\alpha} \langle A', \sigma'\rangle}{\langle A \parallel B, \sigma\rangle \xrightarrow{\alpha} \langle A' \parallel B, \sigma'\rangle} \qquad \textbf{Par}$$

$$\textbf{LR2}\quad \frac{\langle A, \pi_0 \otimes \sigma\rangle \xrightarrow{\alpha} \langle B, \sigma_1\rangle \text{ with } \pi_0 = \pi_\otimes \oplus (\exists_x \pi)_\otimes}{\langle \exists_x^{\pi} A, \sigma\rangle \xrightarrow{\alpha} \langle \exists_x^{\pi\rho} B, \alpha \otimes \sigma \otimes \exists_x \rho\rangle \text{ with } \rho = \sigma_1 \oplus (\alpha \otimes \pi_0 \otimes \sigma)} \text{ for } x \notin sv(\sigma) \cup sv(\alpha) \qquad \textbf{Hide}$$

Table 4. Contextual rules of the labelled semantics for SCCP.

In Table 3 and Table 4 we refine the notion of transition (respectively given in Table 1 and Table 2) by adding a label that carries additional information about the constraints that cause the reduction. Indeed, rules in Table 3 and Table 4 mimic those in Table 1 and Table 2, except for a constraint α that represents the additional information that must be combined with σ in order to fire an action from $\langle A, \sigma\rangle$ to $\langle A', \sigma'\rangle$.

For the rules in Table 3, as well as for rule **LR1**, we can restate the intuition given for their unlabelled counterparts. The difference concerns the axioms for $\textbf{ask}(c)$: if c is not entailed from σ, then some additional information is imported from the environment, ensuring that the state $\alpha \otimes \sigma \leq c$ allows the execution of $\textbf{ask}(c)$.

Once again, the more complex axiom is **LR2**. With respect to **R2**, the additional intuition is that α should not contain the restricted variable x: additional information can be obtained from the environment, as long as it does not interact with data that are private to the local agent. Note that by choosing $\rho = \sigma_1 \oplus (\alpha \otimes \pi_0 \otimes \sigma)$, we are removing α from the update to be memorised in the local store. However, since α is added to the global store, it will not be necessary to receive it again in the future.

Example 6. Consider the agent $A = \exists_x(\textbf{tell}(x \leq 1) \parallel \textbf{ask}(y \leq 5) \mapsto \textbf{stop})$, with the same SLM as in Example 5. We now have the labelled reductions

$$\langle A, \top\rangle \xrightarrow{\mathbf{1}} \langle \exists_x^{[x\leq 1]} \textbf{ask}(y \leq 5) \mapsto \textbf{stop}, \top\rangle \xrightarrow{\alpha} \langle \exists_x^{[x\leq 1, x>1 \vee \alpha]} \textbf{stop}, \top\rangle$$

for every $\alpha \leq (y \leq 5) \oplus (x \leq 1) = (x > 1) \vee (y \leq 5)$ such that $x \notin sv(\alpha)$, e.g., $y \leq 5$. Indeed, for the first reduction we first apply **LA1**, then **LR1**, and finally **LR2**, while for the second reduction we first apply **LA2** and then **LR2**. Looking at the application of **LR2** to the first reduction, we have that $\pi_\otimes = \top = (\exists_x \pi)_\otimes$, thus $\pi_0 = \top$, $\rho = (x \leq 1) \oplus \top = x \leq 1$ and $\exists_x \rho = \top$. Now, consider the second reduction. In that case we have $\pi_0 = (x \leq 1) \oplus \top = x \leq 1$, $\rho = (\alpha \wedge (x \leq 1)) \oplus (x \leq 1) = x > 1 \vee \alpha$ and $\exists_x \rho = \top$.

Remark 7. Concerning the rule **LA2**, an alternative solution would have been to restrict the possible reductions to the one with the maximal label, that is, $\langle \mathbf{ask}(c) \mapsto A, \sigma \rangle \xrightarrow{c \oplus \sigma} \langle A, (c \oplus \sigma) \otimes \sigma \rangle$. However, as hinted at in Example 6, this might have been restrictive in combination with rule **LR2**. Selecting $\alpha = (x > 1) \vee (y \leq 5)$ is problematic, since x occurs free. Instead, the choice of $\alpha = y \leq 5$, or any other value such as $y \leq 4$, $y \leq 3$, ..., fits the intuition of information from the environment triggering the reduction.

Note instead that the choice of removing the requirement $x \notin sv(\alpha)$ and put $\exists_x \alpha$ as label in the conclusion of rule **LR2** would be too liberal. Once again, it would be counterintuitive for the previous example, since $\exists_x((x > 1) \vee (y \leq 5)) = \top$. Or consider the configuration $\gamma = \langle \exists_x^{[x \leq 1]} \mathbf{ask}(x = 0) \mapsto \mathbf{stop}, \top \rangle$: such a configuration should intuitively be deadlocked. However, we have that $\langle \mathbf{ask}(x = 0) \mapsto \mathbf{stop}, x \leq 1 \rangle \xrightarrow{\alpha} \langle \mathbf{stop}, \alpha \wedge x \leq 1 \rangle$ for $\alpha \leq (x = 0) \oplus (x \leq 1) = (x > 1) \vee (x = 0) = x \neq 1$, thus allowing the reduction $\gamma \xrightarrow{\top} \langle \exists_x^{[x \leq 1, \alpha \vee x \leq 1]} \mathbf{stop}, \top \rangle$, which clashes with the intuition that receiving information should not enable reductions involving (necessarily) the restricted variable.

Lemma 10 (On labelled monotonicity). *Let $\langle A, \sigma \rangle \xrightarrow{\alpha} \langle B, \rho \rangle$ be a labelled reduction. Then $\rho = (\rho \oplus (\alpha \otimes \sigma)) \otimes \alpha \otimes \sigma$ and $fv(\langle B, \rho \rangle) \subseteq fv(\langle A, \sigma \rangle) \cup sv(\alpha)$.*

Remark 8. We will later prove that if $\mathbb{S}$ is localised and $\alpha \neq \mathbf{1}$ then $\rho \oplus (\alpha \otimes \sigma) = \mathbf{1}$. In other terms, if $\langle A, \sigma \rangle \xrightarrow{\alpha} \langle B, \rho \rangle$ is a labelled reduction and $\alpha \neq \mathbf{1}$, then $\rho = \alpha \otimes \sigma$. Indeed, since $\alpha \neq \mathbf{1}$ its derivation must use the axiom **LA2**. Consider e.g. a labelled reduction $\langle \exists_x^{\pi} A, \sigma \rangle \xrightarrow{\alpha} \langle \exists_x^{\pi \rho} B, \alpha \otimes \sigma \otimes \exists_x \rho \rangle$. If $\alpha \neq \mathbf{1}$, then $\rho = \mathbf{1}$. Indeed, this is the expected behaviour: if an input from the context is needed, there is no contribution by the agent to the local store, hence the update is correctly **1**.

Definition 16. *A configuration is l-reachable if it can be reached by an initial configuration via a sequence of labelled reductions.*

Lemma 11 (On labelled monotonicity, II). *Let $\langle B \parallel \exists_x^{\pi} C, \sigma \rangle$ be an l-reachable configuration. Then $\sigma = (\sigma \oplus (\exists_x \pi)_{\otimes}) \otimes (\exists_x \pi)_{\otimes}$.*

6 Semantics correspondence and labelled bisimilarity

We collect further formal results in two different subsections: Section 6.1 proves the correspondence between the unlabelled and the labelled semantics, while Section 6.2 proposes a bisimilarity reduction for the labelled semantics.

6.1 On the correspondence between reduction semantics

This section shows the connection between labelled and unlabelled reduction semantics.

Proposition 2 (Soundness). *If $\langle A, \sigma \rangle \xrightarrow{\alpha} \langle B, \sigma' \rangle$ then $\langle A, \alpha \otimes \sigma \rangle \rightarrow \langle B, \sigma' \rangle$.*

The theorem above can be easily reversed, saying that if a configuration $\langle A, \sigma \rangle$ is reachable, then it is also l-reachable via a sequence of reductions labelled with **1**.

Lemma 12. *If* $\langle A,\sigma\rangle \to \langle B,\sigma'\rangle$ *then* $\langle A,\sigma\rangle \xrightarrow{\mathbf{1}} \langle B,\sigma'\rangle$.

These results also ensure that a configuration is reachable iff it is l-reachable. However, we are interested in a more general notion of completeness, possibly taking into account reductions needing a label. For this, we first need some technical lemmas.

Now, note that the proof of every (labelled) reduction is given by the choice of an axiom and a series of applications of the rules **LR1** and **LR2**. Also, note that if $\langle A,\sigma\rangle \xrightarrow{\alpha} \langle B,\sigma'\rangle$ is a reduction via the axiom **LA1**, then $\alpha = \mathbf{1}$.

Proposition 3 (Completeness, I). *Let* $\langle A,\tau\rangle \xrightarrow{\mathbf{1}} \langle B,\tau'\rangle$ *be a reduction via the axiom* **LA1** *for* $\tau \notin \{\bot,\top\}$. *If* $\mathscr{C}$ *is cancellative then* $\langle A,\sigma\rangle \xrightarrow{\mathbf{1}} \langle B,\sigma'\rangle$ *and* $\tau' \oplus \tau = \sigma' \oplus \sigma$ *for every* $\sigma \notin \{\bot,\top\}$.

Proposition 4 (Completeness, II). *Let* $\langle A,\tau\rangle \xrightarrow{\beta} \langle B,\tau'\rangle$ *be a reduction via the axiom* **LA2** *for* $\tau \notin \{\bot,\top\}$. *If* $\mathscr{C}$ *is localised then* $\tau' = \beta \otimes \tau$ *and if* $\alpha \leq (\beta \otimes \tau) \oplus \sigma$ *then* $\langle A,\sigma\rangle \xrightarrow{\alpha} \langle B,\alpha \otimes \sigma\rangle$ *for every* $\sigma \notin \{\bot,\top\}$.

Clearly $\alpha = (\beta \otimes \tau) \oplus \sigma$ is a possible witness. Note however that it might be that $\beta \otimes \tau \not\leq \alpha \otimes \sigma$ if $\sigma = \bot$, in which case $\alpha = \top$.

6.2 Labelled bisimulation

We now exploit the labelled reductions in order to define a suitable notion of bisimilarity without the upward closure condition. As it occurs with the crisp language [1] and the soft variant with global variables [15], barbs cannot be removed from the definition of bisimilarity because they cannot be inferred by the reductions.

Definition 17 (Strong bisimilarity). *A strong bisimulation is a symmetric relation* R *on configurations such that whenever* $(\langle A,\sigma\rangle,\langle B,\rho\rangle) \in R$

1. *if* $\langle A,\sigma\rangle \downarrow_c$ *then* $\langle B,\rho\rangle \downarrow_c$;
2. *if* $\langle A,\sigma\rangle \xrightarrow{\alpha} \gamma_1$ *then there is* γ_2 *such that* $\langle B,\alpha\otimes\rho\rangle \to \gamma_2$ *and* $(\gamma_1,\gamma_2) \in R$;
3. $(\langle A,\sigma\otimes d\rangle,\langle B,\rho\otimes d\rangle) \in R$ *for all* d *such that* $d \not\leq \mathbf{1}$.

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* $(\gamma_1,\gamma_2) \in R$. *We write* $A \sim B$ *if* $\langle A,\mathbf{1}\rangle \sim \langle B,\mathbf{1}\rangle$.

Note that $\langle A,\sigma\rangle \sim \langle B,\rho\rangle$ implies $\sigma = \rho$, as for saturated bisimilarity. We improved on the feasibility of $\sim$ by requiring that the equivalence is upward closed only whenever the store does not decrease. Note that in some cases, e.g. when $\mathscr{C}$ is absorptive (as in [1]), the clause is vacuous. However, thanks to the correspondence results in Section 6.1, it can be proved upward-closed for all d, and thus it is also a congruence.

Proposition 5. *Let* $\langle A,\sigma\rangle \sim \langle B,\rho\rangle$ *and* $d \in \mathscr{C}$. *If* $\mathscr{C}$ *is cancellative then* $\langle A,\sigma\otimes d\rangle \sim \langle B,\rho\otimes d\rangle$.

As for the unlabelled case (Proposition 1), strong bisimilarity is a congruence.

Proposition 6. *Let* $A \sim B$ *and* $C[\cdot]$ *a context. If* $\mathscr{C}$ *is cancellative then* $C[A] \sim C[B]$.

Finally, we can state the correspondence between our bisimilarity semantics.

Theorem 1. $\sim_s \subseteq \sim$. *Moreover, if* $\mathscr{C}$ *is cancellative, then the equality holds.*

7 Related works

As it is possible to appreciate from the survey in [22], the literature on CCP languages is quite ample. In the following of this section we briefly summarise proposals that consider both local and global stores, and information mobility.

The work that is most related to ours is represented by [1]. Anyhow, the differences are significant: in that work the underlying constraint system is crisp, as it can only deal with hard constraints (which indeed we can do as well). Furthermore, the authors of [1] adopt a cylindric algebra instead of a polyadic one, as introduced in Section 1. Finally, as already noted in Remark 5 in this paper, the use of the local store is different with respect to our approach. Since the monoidal operator is idempotent, in [1] the semantics of the hiding operator is simply presented as $\langle \exists^e_x A, \sigma \rangle \to \langle \exists^{e'}_x B, \sigma \otimes \exists_x e' \rangle$ if $\langle A, e \otimes \exists_x \sigma \rangle \to \langle B, e' \otimes \exists_x \sigma \rangle$. Since we have introduced polyadic operators, with their simpler representation of substitutions, and thus we consider agents up-to α-conversion, we can replace $\exists_x \sigma$ with σ by requiring that $x \notin sv(\sigma)$. Most importantly, in [1] the local store e is used to fire a step that only changes the local store to e', and this change is visible in the global store except for the effect on variable x. However, this rule is intrinsically non-deterministic, since many such e' can exist. Moreover, since we are not idempotent we cannot add the whole e' to both the local and the global stores, but only the "difference" between e' and e at each step.

In [20] the authors describe a *spatial* constraint system with operators to specify information and processes moving from a space to another. Such a language provides for the specification of spatial mobility and epistemic concepts such as belief, utterance and lies: besides local stores for agents (representing belief), it can express the epistemic notion of knowledge by means of a derived spatial operator that specifies global shared information. Differently from this work, our approach focuses on preferences, on the concurrent language on top of the system, and on process equivalences.

The process calculi in [2, 12] provide to agents the use of assertions within π-like processes. A soft language is adopted in [12]: from a variant of π-calculus it inherits *explicit fusions*, i.e. simple constraints expressing name equalities, in order to pass constraints from an agent to another. However, the algebraic structure is neither residuated nor polyadic; in addition, no process-equivalence relation is proposed. In [23, 13] processes can send constraints using communication channels much like in the π-calculus.

A further language that uses π-calculus features to exchange constraints between agents, but this time with a probabilistic semantics, is shown in [10]. A congruence relation and a labelled transition system are also shown in the paper.

In [18] the authors propose an extension of the CCP language with the purpose to model process migration within a hierarchical network. Agents bring their local store when they migrate. In [11] the authors enrich a CCP language with the possibility to share (read/write) the information in the global store, and communicate with other agents (via multi-party or handshake).

All the systems described in this section are based on hard constraints, and they do not consider preferences associated with constraints (except [12], whose algebraic structure is however less general). In addition, a very few proposals formalise process equivalences by providing a deeper investigation of the semantics.

8 Conclusions and further works

With the language we presented in this paper, our goal was to further extend and generalise the family of CCP-based languages . In fact, *i)* with respect to crisp languages we can represent preferences, and thus both hard and soft constraints. Then, *ii)* polyadic operators make it possible to have a compact representation of soft constraints (about this, we point the interested reader to [8]), which in turn can be used in several applications, as in hybrid systems, loop invariant generation, and parameter design of control [9]. Furthermore, *iii)* the polyadic algebra we adopted takes advantage of a residuated POM which allows any partially ordered set of preference values, while $\oplus$ permits to easily compute barbs and remove one constraint (store) from another. The use of a non idempotent operator $\otimes$ for combining constraints led us to redesign the local stores proposed in [1], to add to the global store the information added at each computation step only.

An important issue we are currently working on is to remove the requirement of cancellativeness on the first completeness result between the reduction semantics, hence in the correspondence between the barbed and strong bisimilarities, as well as to remove the closure of the store with $d \not\leq \mathbf{1}$ in the definition of strong bisimilarity. Instead, our proposal could be easily extended in order to describe the weak variant of our bisimulation equivalences, which is the main reason why we introduced barbs directly in this paper. Indeed, in such semantics equivalent configurations may have different stores, and barbs were introduced to address this kind of issues [21].

In the case of "soft languages", the removal of constraints can also be partial, while in case of "crisp languages", constraint tokens can only be entirely removed or left in the store. A *retract* operation could also be directly included in the language syntax, in the style of [7, 12], even if it is not in the scope of this paper.

For the future, we conceive more applicative extensions of the language we designed: while in this paper we focused on its formal definition, semantics, and process equivalence, we can think of application fields concerning epistemic concepts or process migration from node to node, as some of the proposals in Section 7 offer.

Separately from the process algebra focus we developed in this paper, we can also think of defining the class of Polynomial *Soft* Constraint Satisfaction Problems (PSCSPs), as accomplished in [26] with crisp constraints, in order to achieve a similar generalisation with respect to CSPs. Hence, we can implement polynomial constraint satisfaction as a SMT module, where agents can tell constraints and ask for their satisfaction.

References

1. Aristizábal, A., Bonchi, F., Palamidessi, C., Pino, L.F., Valencia, F.D.: Deriving labels and bisimilarity for concurrent constraint programming. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 138–152. Springer (2011)
2. Bengtson, J., Johansson, M., Parrow, J., Victor, B.: Psi-calculi: Mobile processes, nominal data, and logic. In: LICS 2009. pp. 39–48. IEEE Computer Society (2009)
3. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. Journal of the ACM **44**(2), 201–236 (1997)
4. Bistarelli, S., Gadducci, F.: Enhancing constraints manipulation in semiring-based formalisms. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (eds.) ECAI 2006. FAIA, vol. 141, pp. 63–67. IOS Press (2006)

5. Bistarelli, S., Montanari, U., Rossi, F.: Soft concurrent constraint programming. ACM Transactions on Computational Logic **7**(3), 563–589 (2006)
6. Bistarelli, S., Pini, M.S., Rossi, F., Venable, K.B.: From soft constraints to bipolar preferences: modelling framework and solving issues. Experimental and Theoretical Artificial Intelligence **22**(2), 135–158 (2010)
7. Bistarelli, S., Santini, F.: A nonmonotonic soft concurrent constraint language to model the negotiation process. Fundamamenta Informaticae **111**(3), 257–279 (2011)
8. Bonchi, F., Bussi, L., Gadducci, F., Santini, F.: Polyadic soft constraints. In: Alvim, M.S., Chatzikokolakis, K., Olarte, C., Valencia, F. (eds.) The Art of Modelling Computational Systems. LNCS, vol. 11760, pp. 241–257. Springer (2019)
9. Borralleras, C., Lucas, S., Oliveras, A., Rodríguez-Carbonell, E., Rubio, A.: SAT modulo linear arithmetic for solving polynomial constraints. Journal of Automed Reasoning **48**(1), 107–131 (2012)
10. Bortolussi, L., Wiklicky, H.: A distributed and probabilistic concurrent constraint programming language. In: Gabbrielli, M., Gupta, G. (eds.) ICLP 2005. LNCS, vol. 3668, pp. 143–158. Springer (2005)
11. Brim, L., Kretínský, M., Jacquet, J., Gilbert, D.R.: Modelling multi-agent systems as synchronous concurrent constraint processes. Computers and Artificial Intelligence **21**(6) (2002)
12. Buscemi, M.G., Montanari, U.: Open bisimulation for the concurrent constraint pi-calculus. In: Drossopoulou, S. (ed.) ESOP 2008. LNCS, vol. 4960, pp. 254–268. Springer (2008)
13. Díaz, J.F., Rueda, C., Valencia, F.D.: Pi+- calculus: A calculus for concurrent processes with constraints. CLEI Electronic Journal **1**(2) (1998)
14. Gadducci, F., Santini, F.: Residuation for bipolar preferences in soft constraints. Information Processing Letters **118**, 69–74 (2017)
15. Gadducci, F., Santini, F., Pino, L.F., Valencia, F.D.: Observational and behavioural equivalences for soft concurrent constraint programming. Journal of Logic and Algebraic Methods in Programming **92**, 45–63 (2017)
16. Galler, B.A.: Cylindric and polyadic algebras. Proceedings of the American Mathematical Society **8**(1), 176–183 (1957)
17. Giacobazzi, R., Debray, S.K., Levi, G.: A generalized semantics for constraint logic programs. In: FGCS 1992. pp. 581–591. IOS Press (1992)
18. Gilbert, D.R., Palamidessi, C.: Concurrent constraint programming with process mobility. In: Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K., Palamidessi, C., Pereira, L.M., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS, vol. 1861, pp. 463–477. Springer (2000)
19. Golan, J.: Semirings and Affine Equations over Them. Kluwer (2003)
20. Guzmán, M., Haar, S., Perchy, S., Rueda, C., Valencia, F.D.: Belief, knowledge, lies and other utterances in an algebra for space and extrusion. Journal of Logic and Algebraic Methods in Programming **86**(1), 107–133 (2017)
21. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 685–695. Springer (1992)
22. Olarte, C., Rueda, C., Valencia, F.D.: Models and emerging trends of concurrent constraint programming. Constraints **18**(4), 535–578 (2013)
23. Réty, J.: Distributed concurrent constraint programming. Fundamenta Informaticae **34**(3), 323–346 (1998)
24. Sági, G.: Polyadic algebras. In: Andréka, H., Ferenczi, M., Németi, I. (eds.) Cylindric-like algebras and algebraic logic, Bolyai Society Mathematical Studies, vol. 22, pp. 367–389. Springer (2013)
25. Saraswat, V.A., Rinard, M.C., Panangaden, P.: Semantic foundations of concurrent constraint programming. In: Wise, D.S. (ed.) POPL 1991. pp. 333–352. ACM Press (1991)
26. Scott, A.D., Sorkin, G.B.: Polynomial constraint satisfaction problems, graph bisection, and the Ising partition function. ACM Transactions on Algorithms **5**(4), 45:1–45:27 (2009)