Soft Concurrent Constraint Programming with Local Variables *

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

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

Abstract. ???

Keywords: Soft concurrent constraint programming, local variables, observational semantics, polyadic algebras, residuated monoids.

1 Introduction

This work originates from [6], where a first implementation of polyadic constraints was ideated (together with a polynomial representation of soft constraints). However, this paper extends that first attempt with a concurrent constraint language on top of its, and process-equivalence relations. The work is organised as follows: in Sect. 2 we present the necessary background on the algebraic structure needed to model polyadic constraints. Then, Sect **??**,

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. Results are mostly drawn from [8], 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 [7] for a detailed introduction and a comparison with other proposals.

Definition 1 (partial order). *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 (join) semilattice (SL) is a PO such that any non-empty finite subset of A has a least upper bound (LUB).*

* Research partially supported by the MIUR PRIN 2017FTXR7S "IT-MaTTerS".

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

Definition 2 (monoid). *A (commutative) monoid is a triple $\langle A, \otimes, \mathbf{1} \rangle$ such that A is a set, $\otimes : A \times A \to A$ is a commutative and associative function, and $\mathbf{1} \in A$ is the* identity *element, namely, $\forall a \in A. a \otimes \mathbf{1} = a$.*

A partially ordered (semi-lattice) monoid is a 4-tuple $\langle A, \leq, \otimes, \mathbf{1} \rangle$ such that $\langle A, \leq \rangle$ is a PO (SL) and $\langle A, \otimes, \mathbf{1} \rangle$ a monoid.

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

Definition 3 (distributivity). *Let $\langle A, \leq, \otimes, \mathbf{1} \rangle$ be a semi-lattice monoid. It is distributive if for any non-empty finite $X \subseteq A$*

- $\forall a \in A. a \otimes \bigvee X = \bigvee \{a \otimes x \mid x \in X\}$.

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 [9], which are known as *c*-semirings in the soft constraint jargon [2] (see e.g. [3] 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 $\forall a, b \in A. a \otimes b \leq a$.

Example 1. Given a (possibly infinite) set V of variables, two semi-lattice monoids are going to play a key role in the following sections. The first one is the semi-lattice monoid $\mathbb{M}(V) = \langle 2^V_{fin}, \subseteq, \cup, \emptyset \rangle$ of finite sub-sets of V, with the usual order given by sub-set inclusion. For the second one, we start by defining the support of an endofunction $f \colon V \to V$ as the set $sv(f) = \{x \in V \mid f(x) \neq x\}$ and $F(V)$ as the set of functions $f \colon V \to V$ with finite support. The semi-lattice monoid of interest is $\mathbb{F}(V) = \langle F(V), id, \circ, \iota \rangle$ where ι is the identity function, $\circ$ is function composition and id is the discrete ordering on $F(V)$.

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 [9]. 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 monoid (RePO) 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 $\forall a, b, c \in A. b \otimes c \leq a \iff c \leq a \oslash b$.x An ReSL is an RePO 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 an RePO. Then*

- $\forall a, b \in A. a \oslash b = \bigvee \{c \mid b \otimes c \leq a\}$,

In order to ease the verification of the algebraic structure, it is often needed a characterisation of residuation via simpler properties, as the ones given below.

Lemma 2. *Let* $\langle A, \leq, \otimes, \mathbf{1} \rangle$ *be a partially ordered monoid and* $\oslash : A \times A \to A$ *a function. Then* $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ *is an RePO if and only if*

- $\forall a, b \in A. b \otimes (a \oslash b) \leq a \leq (b \otimes a) \oslash b$,
- $\forall a, b, c \in A. a \leq b \implies a \otimes c \leq b \otimes c$ *and* $a \oslash c \leq b \oslash c$.

It is easy to show that in any RePO the $\oslash$ operator is also anti-monotone on the second argument, i.e., $\forall a, b, c \in A. a \leq b \implies c \oslash b \leq c \oslash a$. Other properties are also straightforward, such as $\forall a \in A. \mathbf{1} \leq a \oslash a$, which in turn implies that $\forall a \in A. a \otimes (a \oslash a) = a$ and $\forall a, b \in A. a < b \implies \mathbf{1} \not\leq a \oslash b$, where $a < b$ means $a \leq b$ and $a \neq b$. The latter fact suggests the definition below, which identifies sub-classes of residuated monoids that are suitable for an easier manipulation of constraints (see e.g. [3]).

Definition 5 (localisation / invertibility). *An RePO* $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ *is*

- localised *if* $\forall a, b \in A. a \leq b \implies a \oslash b \leq \mathbf{1}$;
- invertible *if* $\forall a, b \in A. a \leq b \implies b \otimes (a \oslash b) = a$.

Note that if a RePO is localised then $\forall a \in A. a \oslash a = \mathbf{1}$.

Remark 2. Some well-known structures used for soft constraints are the *Fuzzy* ($\langle [0,1], \leq, \min, 1 \rangle$), *Probabilistic* ($\langle [0,1], \leq, \times, 1 \rangle$), and *Tropical* ($\langle \mathbb{R}^+, \geq, +, 0 \rangle$) semirings, for $\geq$ the inverse of the standard order (thus 0 the top of the SL). In all these cases the underlying monoids are both invertible and localised, thus the $\oslash$ operator can be also used to (partially) relax constraints (see again [3]).

Moving to ReSLs, next lemma ensures that residuation implies distributivity.

Lemma 3. *Let* $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ *be an ReSL. Then the underlying SL is distributive.*

Distributivity holds also for the empty set and for infinite sets, if the necessary LUBs exist. Instead, it holds only partially for $\oslash$: this follows directly from the monotonicity of $\oslash$ on the first argument, since it implies that $x \oslash a \leq \bigvee X \oslash a$ for all $x \in X$.

Lemma 4. *Let* $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ *be an ReSL and* $X \subseteq A$ *a finite non-empty set. Then*

- $\forall a \in A. \bigvee \{x \oslash a \mid x \in X\} \leq \bigvee X \oslash a$

Also this inequation holds for the empty set and for infinite sets, if the necessary LUBs exist. Moreover, it also holds that $\bigvee \{a \oslash x \mid x \in X\} \geq a \oslash \bigvee X$, since $\oslash$ is anti-monotone on the second argument.

Proposition 1. *Let $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ be an ReSL. The following are equivalent*

1. $\forall a \in A. a \leq \mathbf{1}$
2. $\forall a \in A. \mathbf{1} \oslash a = \mathbf{1}$
3. $\forall a, b \in A. a \leq b \implies b \oslash a = \mathbf{1}$

There are some important classes of ReSLs such that $\oslash$ is easily proved to be distributive in the first argument, while it is not so with respect to the second argument, not even in the absorptive case.

Lemma 5. *Let $\langle A, \leq, \otimes, \oslash, \mathbf{1} \rangle$ be an ReSL such that $\langle A, \leq \rangle$ is a total order and $X \subseteq A$ a finite non-empty set. Then*

- $\forall a \in A. \bigvee \{x \oslash a \mid x \in X\} = \bigvee X \oslash a$

Example 2. Let n be a positive integer and $[n] = \{0, \ldots, n\}$ the segment of integers from 0 to n. We can now define the (bounded) monoid $\mathbb{M}_n$ as the tuple $\langle [n], \geq, \oplus, \ominus, 0 \rangle$, where $\oplus$ and $\ominus$ are the bounded sum and subtraction, which are given as $m \oplus p = min\{n, m+p\}$ and $m \ominus p = max\{0, m-p\}$.

Now, it can be shown that $\mathbb{M}_n$ is an absorptive ReSL, and since it is a total order, $\ominus$ is distributive on the first argument. However, in general it is not distributive on the second one. Consider an integer m such that $m \neq n$ and the set $\{m, m+1\}$: we then have that $(m+1) \ominus \bigvee \{m, m+1\} = 1$, while instead $\bigvee \{(m+1) \ominus m, (m+1) \ominus (m+1)\} = 0$.

3 An Alternative Proposal for Costraint Manipulation

This section presents our personal take on polyadic algebras for ordered monoids: the standard axiomatisation of e.g. [11] has been completely reworked, in order to be adapted to the constraints formalism. We close the section by offering some preliminary insights on the laws for polyadic operators in residuated monoids.

3.1 Cylindric and Polyadic Operators for Ordered Monoids

We now introduce two families of operators that will be used for modelling variables hiding and substitution, which represent key features in languages for manipulating constraints. One is a well-known abstraction for existential quantifiers, the other an axiomatisation of the notion of substitution, and it is proposed as a weaker alternative to diagonals [12], the standard tool for modelling equivalence in constraint programming.[3]

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

Definition 6 (cylindrification). *A cylindric operator $\exists$ over $\mathbb{S}$ and V ia family of monotone operators $\exists_x : A \to A$ indexed by elements in V such that for all $a, b \in A$ and $x, y \in V$*

[3] "Weaker alternative" here means that diagonals allow for axiomatising substitutions at the expenses of working with complete partial orders: see e.g. [8, Definition 11].

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 $sv(a) = \{x \mid \exists_x a \neq a\}$.*

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.

As for notation, for a function $\sigma : V \to V$ and a set $X \subseteq V$, we denote by $\sigma|_X : X \to V$ the obvious restriction, and by $\sigma^c(X) \subseteq V$ the counter-image of X along σ.

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

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

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

3.2 Cylindric and Polyadic Operators for Residuated Monoids

Both algebraic structures introduced in the previous section are quite standard, even if polyadic operators are less-known in the soft-constraints literature: we tailored their presentation to our needs, and indeed the properties presented in Section **??** appear to be original. It is now time to consider the interaction of such structures with residuation. To this end, in the following we assume that $\mathbb{S}$ is a RePO (see Definition 4).

Lemma 6. *Let $X \subseteq V$ be finite. Then it holds*

- $\forall a, b \in A.\ \exists_x (a \oplus \exists_x b) \leq \exists_x a \oplus \exists_x b$.

Proof.

$$\exists_x b \otimes (a \oplus \exists_x b) \leq a \implies \exists_x (\exists_x b \otimes (a \oplus \exists_x b)) \leq \exists_x a \implies$$
$$\exists_x b \otimes \exists_x (a \oplus \exists_x b)) \leq \exists_x a \implies \exists_x (a \oplus \exists_x b) \leq \exists_x a \oplus \exists_x b$$

□

Remark 3. Looking at the proof above, it is clear that $\exists_x (a \oplus \exists_x b) \leq \exists_x a \oplus \exists_x b$ is actually equivalent to state that $\exists_x (a \otimes \exists_x b) \geq \exists_x a \otimes \exists_x b$.

Similarly, it is easy to show that it holds $\forall a, b \in A.\ \exists_x (\exists_x a \oplus b) \leq \exists_x a \oplus \exists_x b$. A similar result relates residuation and polyadic operators.

Lemma 7. *Let $\sigma \in F(V)$. Then it holds*

- $\forall a,b \in A.\ s_\sigma(a \oplus b) \leq s_\sigma a \oplus s_\sigma b$.

Furhtermore, if σ is invertible, then it holds

- $\forall a,b \in A.\ s_\sigma(a \oplus b) = s_\sigma a \oplus s_\sigma b$.

Proof.

$$a \otimes (b \oplus a) \leq b \implies$$
$$s_\sigma[a \otimes (b \oplus a)] \leq s_\sigma b \implies$$
$$s_\sigma a \otimes s_\sigma(b \oplus a) \leq s_\sigma b \implies$$
$$s_\sigma(b \oplus a) \leq s_\sigma b \oplus s_\sigma a$$

As for the second item, note that σ invertible implies $s_{\sigma^{-1}} s_\sigma a = a = s_\sigma s_{\sigma^{-1}} a$. Then we have:

$$s_\sigma b \otimes (s_\sigma a \oplus s_\sigma b) \leq s_\sigma a \implies$$
$$s_{\sigma^{-1}}(s_\sigma b \otimes (s_\sigma a \oplus s_\sigma b)) \leq s_{\sigma^{-1}} s_\sigma a \implies$$
$$s_{\sigma^{-1}} s_\sigma b \otimes s_{\sigma^{-1}}(s_\sigma a \oplus s_\sigma b) \leq a \implies$$
$$b \otimes s_{\sigma^{-1}}(s_\sigma a \oplus s_\sigma b) \leq a \implies$$
$$s_{\sigma^{-1}}(s_\sigma a \oplus s_\sigma b) \leq a \oplus b \implies$$
$$s_\sigma(s_{\sigma^{-1}}(s_\sigma a \oplus s_\sigma b)) \leq s_\sigma(a \oplus b) \implies$$
$$s_\sigma a \oplus s_\sigma b \leq s_\sigma(a \oplus b)$$

3.3 Polyadic Soft Constraints

We are now ready to advance our proposal of soft constraints; follows yet generalises [4], whose underlying algebraic structure is the one of absorptive 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 ReSL. 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.*

In this section and in the following one, we denote by $\mathscr{C}$ 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 $c\eta$.

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 $c : (V \to D) \to A$ that 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 cardinality 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 forms a ReSL, with the structure lifted from $\mathbb{S}$.

Lemma 8 (the ReSL of constraints). *The ReSL of constraints* $\mathbb{C}$ *is defined as the tuple* $\langle \mathscr{C}, \leq, \otimes, \oplus, \mathbf{1} \rangle$ *such that*

- $c_1 \leq c_2$ *if* $c_1\eta \leq c_2\eta$ *for all* $\eta : V \to D$,
- $(c_1 \otimes c_2)\eta = c_1\eta \otimes c_2\eta$,
- $(c_1 \oplus c_2)\eta = c_1\eta \oplus c_2\eta$,
- $\mathbf{1}\eta = \mathbf{1}$.

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

Lemma 9 (Cylindric and polyadic operators for (soft) constraints). *The ReSL 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* $c \in \mathscr{C}, x \in V$
- $(s_\sigma c)\eta = c(\eta \circ \sigma)$ *for all* $c \in \mathscr{C}, \sigma \in F(V)$

Hiding means eliminating variables from the support: $supp(\exists_x c) \subseteq supp(c) \setminus x$.[4]

Proof. The properties of the pomonoid action (see Definition **??**) are easily shown to hold for both operators. As for the cylindric laws (see Definition 6), first note that the set of functions ρ such that $\eta \mid_{V\setminus\{x\}} = \rho \mid_{V\setminus\{x\}}$ is actually finite. Thus, we have that

$$\begin{aligned}
(\exists_x(c \otimes \exists_x d))\eta &= \bigvee_\rho \{(c \otimes \exists_x d)\rho \mid \eta \mid_{V\setminus\{x\}} = \rho \mid_{V\setminus\{x\}}\} \\
&= \bigvee_\rho \{c\rho \otimes (\exists_x d)\rho \mid \eta \mid_{V\setminus\{x\}} = \rho \mid_{V\setminus\{x\}}\} \\
&= \bigvee_\rho \{c\rho \otimes (\bigvee_\xi \{d\xi \mid \rho \mid_{V\setminus\{x\}} = \xi \mid_{V\setminus\{x\}}\}) \mid \eta \mid_{V\setminus\{x\}} = \rho \mid_{V\setminus\{x\}}\} \\
&= \bigvee_\rho \{c\rho \mid \eta \mid_{V\setminus\{x\}} = \rho \mid_{V\setminus\{x\}}\} \otimes \bigvee_\xi \{d\xi \mid \eta \mid_{V\setminus\{x\}} = \xi \mid_{V\setminus\{x\}}\} \\
&= (\exists_x c)\eta \otimes (\exists_x d)\eta
\end{aligned}$$

[da rivedere] Let us now move to the polyadic laws (see Definition 7).We just consider the third item, and we assume that $\sigma \mid_{\sigma^c(X)}$ is injective, thus

$$\begin{aligned}
(\exists_X s_\sigma c)\eta &= \bigvee_\rho \{(s_\sigma c)\rho \mid \eta \mid_{V\setminus X} = \rho \mid_{V\setminus X}\} = \bigvee_\rho \{c(\rho \circ \sigma) \mid \eta \mid_{V\setminus X} = \rho \mid_{V\setminus X}\} \\
&= \bigvee_\xi \{c\xi \mid (\eta \circ \sigma) \mid_{V\setminus\sigma^c(X)} = \xi \mid_{V\setminus\sigma^c(X)}\} \\
&= (\exists_{\sigma^c(X)} c)(\eta \circ \sigma) = (s_\sigma \exists_{\sigma^c(X)} c)\eta
\end{aligned}$$

where it always holds that $\eta \mid_{V\setminus X} = \rho \mid_{V\setminus X}$ implies $(\eta \circ \sigma) \mid_{V\setminus\sigma^c(X)} = (\rho \circ \sigma) \mid_{V\setminus\sigma^c(X)}$, while since $\sigma \mid_{\sigma^c(X)}$ is injective we have that a ξ satisfying $(\eta \circ \sigma) \mid_{V\setminus\sigma^c(X)} = \xi \mid_{V\setminus\sigma^c(X)}$ can be decomposed as $\rho \circ \sigma$ for a ρ such that $\eta \mid_{V\setminus X} = \rho \mid_{V\setminus X}$ (otherwise, it could happen that for some $\{x, y\} \subseteq \sigma^c(X)$ we have that $\sigma(x) = \sigma(y)$ and $\xi(x) \neq \xi(y)$). □

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

Note also that the diagonal elements are not guaranteed to be $\otimes$-compact, even if they have finite support, since $\top$ is not necessarily so. To this end, we close the section by adding the simple result below to the soft constraint lore.

Proposition 2. *Let $c \in \mathbb{C}$ be a constraint. It is $\otimes$-compact if and only if it has finite support and $c\eta$ is $\otimes$-compact for all η.*

4 Polyadic Soft CCP: language and reduction/barbed semantics

This section introduces our (meta-)language. We fix a set of variables V, ranged over by $x, y, \ldots$, and an invertible CLIM $\mathbb{S} = \langle \mathscr{C}, \leq, \otimes \rangle$, which is polyadic over V and whose elements are ranged over by $c, d, \ldots$.

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

$$A ::= \mathbf{stop} \mid \mathbf{tell}(c) \mid \mathbf{ask}(c) \to A \mid A \parallel A \mid p(x) \mid \exists_x^{\rho} A.$$

We consider here the extended agent $\exists_x^{\rho} A$, where ρ is meant to represent a local store. More precisely, the extended agent allows to carry some information about the hidden variable x in an incremental way. In the following, we will often write $\exists_x A$ for $\exists_x^{\mathbf{1}} A$

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

We now move to consider the reduction semantics of our language.

Definition 10 (Substitutions). *Let ${}^{y}/_{x} \in F(V)$, defined as:*

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

. We define the substitution operator ${}^{y}/_{x}] : \mathscr{A} \to \mathscr{A}$ on agents as follows:

- $\mathbf{stop}{}^{y}/_{x}] = \mathbf{stop}$
- $\mathbf{tell}(c){}^{y}/_{x}] = \mathbf{tell}(s_{{}^{y}/_{x}]} c)$
- $p(w){}^{y}/_{x}] = p({}^{y}/_{x}](w))$
- $(\mathbf{ask}(c) \to A){}^{y}/_{x}] = \mathbf{ask}(s_{{}^{y}/_{x}]}(c)) \to A{}^{y}/_{x}]$
- $(\exists_w^{\rho} A){}^{y}/_{x}] = \exists_w^{(s_{{}^{y}/_{x}]}\rho)} A{}^{y}/_{x}]$ *for* $w \notin \{x, y\}$
- $(A_1 \parallel A_2){}^{y}/_{x}] = (A_1{}^{y}/_{x}] \parallel A_2{}^{y}/_{x}])$

Remark 4. In the following, we consider terms to be equivalent up to α-conversion, meaning that terms which differ only for hidden variables can be considered equivalent:

$$\exists_x^{\rho} A = \exists_y^{(s_{{}^{y}/_{x}]}\rho)} A{}^{y}/_{x}] \quad for \;\; y \notin sv(\rho) \cup fv(A)$$

Note that this holds thanks to the underlying monoid properties, where we have that $\exists_x \sigma = \exists_y s_{{}^{y}/_{x}]}(\sigma)$ if $y \notin sv(\sigma)$.

Thanks to α-conversion, we can define substitution for $\exists^{\rho}_{x}A$ as above, even if the substitution function $[^{y}/_{x}]$ is not injective. Other cases can be managed by renaming, since the following holds:

Proposition 3. *Let $A \in \mathscr{A}$, $x \notin fv(A)$. Then $A[^{y}/_{x}] = A$.*

Proof. For **stop**, **tell**(c), $p(w)$ the statement is trivially true, as we have that

- **stop**$[^{y}/_{x}] =$ **stop**
- **tell**$(c)[^{y}/_{x}] =$ **tell**$(s_{[^{y}/_{x}]}c) =$ **tell**(c) by polyadic laws and $x \notin fv($**tell**$(c)) = sv(c)$.
- $p(w)[^{y}/_{x}] = p(w)$ by $sv(p(w)) = w \wedge x \notin sv(p(w)) \implies w \neq x$.

As for the other agents, the statement can be proved by inductive reasoning, assuming it to be true for subagents in **ask**$(c) \to A$, $(A_1||A_2)$ and $\exists^{\rho}_{w}A$. Indeed we obtain:

- $($**ask**$(c) \to A)[^{y}/_{x}] =$ **ask**$(s_{[^{y}/_{x}]}c) \to A[^{y}/_{x}] =$ **ask**$(c) \to A$ by hypothesis and polyadic laws.
- $(A_1||A_2)[^{y}/_{x}] = (A_1[^{y}/_{x}] \parallel A_2[^{y}/_{x}])) = (A_1 \parallel A_2)$ by hypothesis.

As for $\exists^{\rho}_{w}A$, recall that $fv(\exists^{\rho}_{w}A) = (fv(A) \cup sv(\rho)) \setminus \{w\}$, thus $x \neq w \wedge x \notin fv(\exists^{\rho}_{w}A) \implies x \notin fv(A) \wedge x \notin sv(\rho)$. Then we have two possible cases:

- $w \neq x$. $(\exists^{\rho}_{w}A)[^{y}/_{x}] = \exists^{s_{[^{y}/_{x}]}\rho}_{w}A[^{y}/_{x}] = \exists^{\rho}_{w}A[^{y}/_{x}]$ (by $x \neq w$) $= \exists^{\rho}_{w}A$ (by hypotheses).
- $w = x$. $(\exists^{\rho}_{x}A)[^{y}/_{x}] = (((\exists^{\rho}_{z}A)[^{z}/_{x}])[^{y}/_{x}]$ for $z \notin sv(\rho) \cup fv(A) = (\exists_{z}^{s_{[^{y}/_{x}]}(s_{[^{z}/_{x}]}\rho)}(A[^{z}/_{x}])[^{y}/_{x}]) = (\exists^{(s_{[^{z}/_{x}]}\rho)}_{z}A[^{z}/_{x}])$by $x \notin sv(c) \cup fv(A[^{z}/_{x}]) \equiv_{\alpha} \exists^{\rho}_{x}A$.

Definition 11 (Reductions). *Let $\Gamma = \mathscr{A} \times \mathscr{C}$ be the set of* configurations. *The* direct reduction semantics *for SCCP is the pair $\langle \Gamma, \mapsto \rangle$ such that $\mapsto \subseteq \Gamma \times \Gamma$ is the family of binary relations indexed over sets of variables, i.e., $\mapsto = \bigcup_{\Delta \subseteq V} \mapsto_{\Delta}$ and $\mapsto_{\Delta} \subseteq \Gamma \times \Gamma$, obtained by the rules in Table 1.*

The reduction semantics *for SCCP is the pair $\langle \Gamma, \to \rangle$ such that $\to \subseteq \Gamma \times \Gamma$ is the family of binary relations indexed over sets of variables, i.e., $\to = \bigcup_{\Delta \subseteq V} \to_{\Delta}$ and $\to_{\Delta} \subseteq \Gamma \times \Gamma$, obtained by the rules in Table 1 and Table 2.*

A1 $\langle$**tell**$(c), \sigma\rangle \mapsto \langle$**stop**$, \sigma \otimes c\rangle$ **Tell**

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

A3 $\dfrac{p(x) = A \in \mathscr{P}}{\langle p(y), \sigma\rangle \mapsto \langle A[^{y}/_{x}], \sigma\rangle}$ **Rec**

A4 $\dfrac{\langle A, \rho \otimes \sigma_0\rangle \mapsto \langle B, \sigma_1\rangle \wedge x \notin sv(\sigma) \wedge \sigma_0 = \sigma \oplus \exists_x \rho}{\langle \exists^{\rho}_{x}A, \sigma\rangle \mapsto \langle \exists_{x}^{\sigma_1 \oplus \sigma_0}B, \sigma_0 \otimes \exists_x(\sigma_1 \oplus \sigma_0)\rangle}$ **Hide**

Table 1. Axioms of the reduction semantics for SCCP.

$$\textbf{R1}\ \frac{\langle A,\sigma\rangle \rightarrow_\Delta \langle A',\sigma'\rangle \wedge fv(B)\subseteq\Delta}{\langle A \parallel B,\sigma\rangle \rightarrow_\Delta \langle A' \parallel B,\sigma'\rangle}\ \textbf{Par1}$$

$$\textbf{R2}\ \frac{\langle A,\sigma\rangle \rightarrow_\Delta \langle A',\sigma'\rangle \wedge fv(B)\subseteq\Delta}{\langle B \parallel A,\sigma\rangle \rightarrow_\Delta \langle B \parallel A',\sigma'\rangle}\ \textbf{Par2}$$

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

The split distinguishes between axioms and rules guaranteeing the closure with respect to the parallel operator. Indeed, rules **R1** and **R2** model the interleaving of two agents in parallel. 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:

In axiom **A4** we consider instead local variables, where the hiding operator carries some information on the variables it abstracts. More precisely, according to [5] we consider an extended operator $\exists_x^\rho$, for ρ the local store. Thanks again to the residuation operator, the rule for the extended hidings can be defined as **Hide-local**. The precondition states that $x \notin sv(\sigma)$: this is not restrictive, due to α-equivalence. Furthermore, it ensures that σ_0 does not contain free occurences of x, as it is obtained by removing $\exists_x\rho$ from σ. The intuition is then that x can only appear in the local store ρ, and then in σ_1. Thus, hiding x means to remove it from the resulting store, by letting it $\sigma_0 \otimes \exists_x(\sigma_1 \oplus \sigma_0)$. On the other hand, the local information carried by the hide operator is updated to $\sigma_1 \oplus \sigma_0$, which can still contain x.

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 substitution $[^z/_w]$.

Lemma 10 (On monotonicity). *Let $\langle A,\sigma\rangle \rightarrow \langle B,\sigma'\rangle$ be a reduction. Then*

1. $sv(\sigma') \subseteq sv(\sigma)\cup fv(A) \subseteq \Delta$;
2. $\sigma \leq \sigma'$;
3. $\langle A,\sigma\rangle \rightarrow_{\Delta'} \langle B,\sigma'\rangle$ *for all* $\Delta'.sv(\sigma)\cup fv(A)\subseteq\Delta' \wedge fv(B)\cap\Delta' \subseteq fv(A)$;
4. $\langle A,\sigma\otimes\rho\rangle \rightarrow_\Delta \langle B,\sigma'\otimes\rho\rangle$ *for all* $\rho \in \mathscr{C}.sv(\rho)\subseteq\Delta$.

All statements are straightforward.

As for item 1, by construction $sv(\sigma)\cup fv(A)\subseteq\Delta$: since only rule **A1** can modify the store and $sv(\sigma\otimes c)\subseteq sv(\sigma)\cup sv(c)$, then the statement holds. Similarly for item 2, since $\sigma \leq \sigma\otimes c$. Item 3 is again true by construction. As for item 4, it suffices to also note that $\sigma,\rho \in \mathscr{C}$ ensure that $\sigma\otimes\rho \in \mathscr{C}$ and clearly **A2** will still be executable.

5 Saturated Bisimulation

As proposed in [1] for crisp languages, we define a barbed equivalence between two agents [10]. 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 12 (Barbs). *Let $\langle A, \sigma \rangle$ be a configuration and $c \in \mathscr{C}$ and we say that $\langle A, \sigma \rangle$ verifies c, or that $\langle A, \sigma \rangle \downarrow_c$ holds, if $\sigma \leq c$.*

However, since *barbed bisimilarity* is an equivalence already for CCP, along [1] we propose the use of *saturated bisimilarity* in order to obtain a congruence: Definition 13 and Definition 15 respectively provide the strong and weak definition of saturated bisimilarity.

Definition 13 (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 $\langle A, \sigma \rangle \longrightarrow \gamma_1'$ then there is γ_2' such that $\langle B, \rho \rangle \longrightarrow \gamma_2'$ and $(\gamma_1', \gamma_2') \in R$;*
3. *$(\langle A, \sigma \otimes d \rangle, \langle B, \rho \otimes d \rangle) \in R$ for all $d \in \mathscr{C}$.*

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 $\langle A, \bot \rangle \sim_s \langle B, \bot \rangle$.

We now let $\longrightarrow^*$ denote the reflexive and transitive closure of $\longrightarrow$, restricted to increasing computations.

Definition 14 (Weak barbs). *Let $\langle A, \sigma \rangle$ be a configuration and $c \in \mathscr{C}$. We say that $\langle A, \sigma \rangle$ weakly verifies c, or that $\langle A, \sigma \rangle \Downarrow_c$ holds, if there exists $\gamma' = \langle B, \rho \rangle$ such that $\gamma \longrightarrow^* \gamma'$ and $c \leq \exists_X \rho$ for $X = fv(\gamma') \setminus fv(\gamma)$.*

Definition 15 (Weak saturated bisimilarity). *A weak 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 $\langle A, \sigma \rangle \longrightarrow \gamma_1'$ then there is γ_2' such that $\langle B, \rho \rangle \longrightarrow^* \gamma_2'$ and $(\gamma_1', \gamma_2') \in R$;*
3. *$(\langle A, \sigma \otimes d \rangle, \langle B, \rho \otimes d \rangle) \in R$ for all $d \in \mathscr{C}$.*

We say that γ_1 and γ_2 are weakly saturated bisimilar ($\gamma_1 \approx_s \gamma_2$) if there exists a weak saturated bisimulation R such that $(\gamma_1, \gamma_2) \in R$. We write $A \approx_s B$ if $\langle A, \bot \rangle \approx_s \langle B, \bot \rangle$.

The asymmetry is functional to later sections. However, it is clearly equivalent to the standard symmetric version.

Definition 16 (Weak saturated bisimilarity, 2). *Weak saturated bisimilarity coincides with the relation obtained from Definition 13 by replacing $\longrightarrow$ with $\longrightarrow^*$ and $\downarrow_c$ with $\Downarrow_c$.*

Since $\sim_s$ (and $\approx_s$) is a saturated bisimulation, it is clearly upward closed and it is also a congruence: indeed, a context can modify the behaviour of a configuration only by adding constraints to its store.

6 Labelled transition system

Although $\approx_s$ is fully abstract, it is somewhat unsatisfactory because of the upward-closure, i.e., the quantification in condition *3* of Definition 15.

Definition 17 (Labelled reductions). *Let $\Gamma = \mathscr{A} \times \mathscr{C}$ 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 $\langle \mathscr{C}, 2^V \rangle$, i.e., $\mapsto = \bigcup_{\alpha \in \mathscr{C}, \Delta \subseteq V} \overset{A}{\mapsto}$ and $\overset{A}{\mapsto} \subseteq \Gamma \times \Gamma$, obtained by the rules in Table 3.*

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 $\langle \mathscr{C}, 2^V \rangle$; $\overset{A}{\rightarrow} \subseteq \Gamma \times \Gamma$ can obtained by the rules in Table 3 and Table 4.*

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. Hence, we define a new labelled transition system (LTS) obtained by the family of relations $\overset{\alpha}{\longmapsto} a \subseteq \Gamma \times \Gamma$ indexed over $\langle \mathscr{C}, 2^V \rangle$; Rules in Table 3 and Table 4 are identical to those in Table 1 and Table 2, 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., $\langle A, \sigma \otimes \alpha \rangle \longrightarrow \langle A', \sigma' \rangle$.

LA1	$\langle \mathbf{tell}(c), \sigma \rangle \overset{\bot}{\longmapsto} \langle \mathbf{stop}, \sigma \otimes c \rangle$	**Tell**
LA2	$\langle \mathbf{ask}(c) \rightarrow A, \sigma \rangle \overset{c \oplus \sigma}{\longmapsto} \langle A, \sigma \otimes (c \oplus \sigma) \rangle$	**Ask**
LA3	$\dfrac{p(x) = A \in \mathscr{P}}{\langle p(y), \sigma \rangle \overset{\bot}{\longmapsto} \langle A[^y/_x], \sigma \rangle}$	**Rec**
LA4	$\dfrac{\langle A, c \otimes d_0 \rangle \overset{\alpha}{\mapsto} \langle B, d_1 \rangle \wedge d_0 = (d \otimes \alpha) \oplus \exists_x c \wedge x \notin sv(d) \cup sv(c) \wedge c_1 = d_1 \oplus d_0}{\langle \exists_x^c A, d \rangle \overset{\alpha}{\mapsto} \langle \exists_x^{c_1} B, d_0 \otimes \exists_x c_1 \rangle}$	**Hide**

Table 3. Axioms of the labelled semantics for SCCP.

LR1	$\dfrac{\langle A, \sigma \rangle \overset{\alpha}{\rightarrow} \langle A', \sigma' \rangle \wedge fv(B) \subseteq \Delta}{\langle A \parallel B, \sigma \rangle \overset{\alpha}{\rightarrow} \langle A' \parallel B, \sigma' \rangle}$	**Par1**
LR2	$\dfrac{\langle A, \sigma \rangle \overset{\alpha}{\rightarrow} \langle A', \sigma' \rangle \wedge fv(B) \subseteq \Delta}{\langle B \parallel A, \sigma \rangle \overset{\alpha}{\rightarrow} \langle B \parallel A', \sigma' \rangle}$	**Par2**

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

7 Semantics equivalence

Theorem 1 (Soundness). $\langle A, a\rangle \overset{c}{\mapsto} \langle B, b\rangle \implies \langle A, a \otimes c\rangle \mapsto \langle B, b\rangle$

Proof. Tell and Rec

$$\sigma \otimes \bot = \sigma$$

$$\langle \mathbf{tell}(c), \sigma\rangle \overset{\bot}{\mapsto} \langle \mathbf{stop}, \sigma\rangle \implies$$

$$\langle \mathbf{tell}(c), \sigma \otimes \bot\rangle = \langle \mathbf{tell}(c), \sigma\rangle \mapsto \langle stop, \sigma\rangle$$

$$\langle p(y), \sigma\rangle \overset{\bot}{\mapsto} \langle A[^{y}/_{x}], \sigma\rangle \implies$$

$$\langle p(y), \sigma \otimes \bot\rangle = \langle p(y), \sigma\rangle \mapsto \langle A[^{y}/_{x}], \sigma$$

Ask

$$\langle \mathbf{ask}(c) \to A, \sigma\rangle \overset{c \oplus \sigma}{\longmapsto} \langle A, \sigma \otimes (c \oplus \sigma)\rangle \implies$$

NB $\sigma \otimes (c \oplus \sigma) \leq c$

$$\langle \mathbf{ask}(c) \to A, \sigma \otimes (c \oplus \sigma)\rangle \mapsto \langle A, \sigma \otimes (c \oplus \sigma)\rangle$$

Hide
hyp: $\langle A, c \otimes d_0\rangle \overset{\alpha}{\mapsto} \langle B, d_1\rangle \implies \langle A, c \otimes d_0 \otimes \alpha\rangle \mapsto \langle B, d_1\rangle$
$d_0 = (d \otimes \alpha) \oplus \exists_x c$
$c_1 = d_1 \oplus d_0$

$$\langle \exists^{c}_{x} A, d\rangle \overset{\alpha}{\mapsto} \langle \exists^{d_1 \oplus ((d\otimes\alpha)\oplus\exists_x c)} B, ((d \otimes \alpha) \oplus \exists_x c) \otimes \exists_x (d_1 \otimes ((d \otimes \alpha) \oplus \exists_x c)) \implies$$

$$\langle \exists^{c}_{x} A, d \otimes \alpha\rangle \mapsto \langle \exists^{d_1 \oplus ((d\otimes\alpha)\oplus\exists_x c)} B, ((d \otimes \alpha) \oplus \exists_x c) \otimes \exists_x (d_1 \otimes ((d \otimes \alpha) \oplus \exists_x c))$$

Theorem 2 (Completeness). $c \leq a?$, $\langle A, a\rangle \mapsto \langle B, b\rangle \implies \langle A, c\rangle \overset{\alpha}{\mapsto} \langle B, d\rangle$ *with* $c \otimes \alpha \otimes \beta = a \wedge d \otimes \beta = b$ *for some* β.

Proof. Tell and Rec

$$\langle \mathbf{tell}(c), \sigma\rangle \mapsto \langle \mathbf{stop}, \sigma\rangle \wedge (c \leq a)$$

$$\langle \mathbf{tell}(c), \sigma'\rangle \overset{\bot}{\mapsto} \langle \mathbf{stop}, \sigma'\rangle$$

$$\implies \beta = \sigma \oplus \sigma' = \bigvee \{c \mid \sigma' \otimes c \leq \sigma\}$$

8 Concluding Remarks

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. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. Journal of ACM 44(2), 201–236 (1997)
3. 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)
4. Bistarelli, S., Montanari, U., Rossi, F.: Soft concurrent constraint programming. ACM Transactions on Computational Logic 7(3), 563–589 (2006)
5. de Boer, F.S., Gabbrielli, M., Marchiori, E., Palamidessi, C.: Proving concurrent constraint programs correct. Transactions on Programming Languages and Systems 19(5), 685–725 (1997)
6. 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: A Journey from Logic and Concurrency to Security and Privacy - Essays Dedicated to Catuscia Palamidessi on the Occasion of Her 60th Birthday. Lecture Notes in Computer Science, vol. 11760, pp. 241–257. Springer (2019)
7. Gadducci, F., Santini, F.: Residuation for bipolar preferences in soft constraints. Information Processing Letters 118, 69–74 (2017)
8. Gadducci, F., Santini, F., Pino, L.F., Valencia, F.D.: Observational and behavioural equivalences for soft concurrent constraint programming. Logical and Algebraic Methods in Programming 92, 45–63 (2017)
9. Golan, J.: Semirings and Affine Equations over Them. Kluwer (2003)
10. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 685–695. Springer (1992)
11. 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)
12. 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)