# Soft Constraint Automata with Memory

Kasper Dokter[1], Fabio Gadducci[2], and Francesco Santini[3(✉)]

[1] Centrum Wiskunde & Informatica, Amsterdam, Netherlands
K.P.C.Dokter@cwi.nl
[2] Dipartimento di Informatica, University of Pisa, Pisa, Italy
fabio.gadducci@unipi.it
[3] Dipartimento di Matematica e Informatica, University of Perugia, Perugia, Italy
francesco.santini@unipg.it

**Abstract.** In this paper, we revise the notion of *Soft Constraint Automata*, where automata transitions are weighted and consequently each action is associated with a preference value. We first relax the underlying algebraic structure that models preferences, with the purpose to use bipolar preferences (i.e., both positive and negative ones). Then, we equip automata with memory cells, that is, with an internal state to remember and update information from transition to transition. Finally, we revise automata operators, as join and hiding.

**Keywords:** Constraint automata with memory · Soft constraints

## 1 Introduction

In the history of Computer Science, many coordination languages have been proposed for the specification and implementation of interaction protocols, in order to let software components communicate. Such formalisms include process calculi, concurrent objects, actors, agents, shared memory, message passing, and more. A distinctive feature of these formalisms is that they are all primarily action-based models that provide constructs for the direct specification of things that interact, rather than a direct specification of interaction (i.e., protocols).

This is one of the main motivations behind the long-running success of the Reo language [2], whose distinctive feature instead is to treat interaction as an explicit first-class concept. Reo comes with its own composition operators, and it allows for specifying more complex interaction protocols by combining simpler, and possibly primitive, protocols. In practice, Reo connectors impose constraints on the order in which the components can exchange data items with each other; even though the basic primitive channels are simple, Reo connectors can actually describe rather complex protocols.

The literature offers tens of different semantics formalisms to express the behaviour of Reo connectors [17]: *co-algebraic*, *colouring*, and other models based on, for instance, constraints and Petri nets. The *operational* models (i.e., automata) are probably the most popular approaches: *Constraint Automata* [8]

(*CA*) and (several) related variants, and *Context-sensitive Automata*. Variants of CA consists in *Timed*, *Probabilistic*, *Continuous-time*, *Quantitative Resource-sensitive timed*, and *Transactional* extensions.

In the remainder of this paper, our aim is to both relax and extend one of the chronologically latter variants of CA, hence not included in the survey in [17]: *Soft Constraint Automata* [6], also called *Soft Component Automata* [19, 20] (SCA in both cases). SCA is a state-transition system where transitions are labelled with actions and preferences. Higher-preference transitions typically contribute more towards the goal of the component.

The aim of this paper consists of three sub-goals. First, we relax the definition of the underlying structure that models preferences: instead of semirings as in [6,19,20], we exploit partially ordered monoids (see Sect. 2) to naturally relax soft constraints in order to represent bipolar preferences as labels for automata transitions. In this way, we can express both positive and negative values, e.g., costs and retributions for firing a transition rule.

Second, we extend SCA with a notion of memory (SCAM), as Arbab and co-authors have already accomplished for CA [18]. Each transition of a SCAM can also put a condition on the current data assigned to a finite set of *memory cells*, and update their respective values. Therefore, together with states, memory cells determine the configuration of a connector, and thus can influence the observed behaviour of the component.

The third and last intention is less scientific, but more significant from our side: we feel the need to celebrate Farhad's influential intuition behind Reo, and his far-reaching contribution to many fields of Computer Science as, to name only two of them, Concurrency Theory and Coordination Models and Languages. Indeed, besides personal gratitude,[1] our will is to continue playing with *"Puff's gigantic tail"* for a long time ahead [3].

The outline of the paper is as follows: in Sect. 2 we set the background notions behind the algebraic structure we use for our soft constraints, that is partially ordered monoids; then, in Sect. 3 we just define soft constraint functions. In Sect. 4 we introduce and formally define SCAM, while Sect. 5 summarises the related work. Finally, Sect. 6 wraps up the paper with conclusive thoughts and hints about related future research.

## 2   Partially Ordered Monoids

The first step is to define an algebraic structure for modelling preferences. It falls into the range of *bipolar* approaches: we refer to [14] for the missing proofs as well as for an 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 PO is a complete lattice (CL) if any subset of $A$ has a least upper bound (LUB).*

The LUB of a subset $X \subseteq A$ is denoted $\bigvee X$, and it is unique. By definition $\bot = \bigvee \emptyset$ is the bottom of the PO and $\top = \bigvee A$ is the top.

**Definition 2 (PO monoids).** *A (commutative) monoid is a triple $\langle A, \oplus, \mathbf{0} \rangle$ such that $\oplus : A \times A \to A$ is a commutative and associative function satisfying*

 – *$\forall a \in A. a \oplus \mathbf{0} = a$, where $\mathbf{0} \in A$ is the* identity *element.*

*A partially ordered monoid (POM) is a 4-tuple $\langle A, \leq \oplus, \mathbf{0} \rangle$ such that $\langle A, \leq \rangle$ is a PO and $\langle A, \oplus, \mathbf{0} \rangle$ a monoid. It is monotone if*

 – *$\forall a, b, c \in A. a \leq b \implies a \oplus c \leq b \oplus c$*

*and it is distributive if*

 – *$\forall a \in A. \forall X \subseteq A. a \oplus \bigvee X = \bigvee \{a \oplus x \mid x \in X\}$.*

*whenever $X$ is finite. A complete lattice monoid (CLM) is a POM such that the underlying order is a CL, it is monotone if the underlying POM is so and it is distributive if the property holds for possibly infinite subsets.*

Note that in a distributive POM the $\oplus$ operation is monotone. In the following, we usually use an infix notation $a \oplus b$ for $\oplus(a, b)$.

*Remark 1.* It is now easy to show that distributive POMs are *tropical* semirings, i.e., semirings with a sum operator $a \oplus b = \bigvee\{a, b\}$ that is idempotent. If $\mathbf{0}$ is also the top of the PO we end up in what are called *absorptive* semirings [16] in the algebra literature, which in turn are known as *c*-semirings in the soft constraint jargon [10]. Combined with monotonicity, imposing $\mathbf{0}$ to be the top means that preferences are negative, i.e., $\forall a, b \in A. a \oplus b \leq a$. Indeed, most better known structures that are used in the soft constraints literature are absorptive semirings whose underlying, distributive POM is actually a CLM: among them we recall the *Boolean* ($\langle \{false, true\}, \to, \wedge, true \rangle$), *Fuzzy* ($\langle [0, 1], \leq, \min, 1 \rangle$), *Probabilistic* ($\langle [0, 1], \leq, \times, 1 \rangle$), and *Tropical* ($\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +, 0 \rangle$) semirings (where, in the latter, $\geq$ the inverse of the standard order, thus $+\infty$ is the bottom and 0 the top of the CLM, respectively).

*Remark 2.* Note that CLMs feature an operator that carries the intuitive meaning of subtraction, which can be defined as $a \ominus b = \bigvee\{c \mid b \oplus c \leq a\}$. It satisfies the usual property of residuation, namely $b \oplus c \leq a \iff c \leq a \ominus b$: see e.g. [11] for a brief survey on residuation for absorptive semirings.

### 2.1 Cylindric Algebras

We now introduce two families of operators that we use for modelling suitable operators on automata. They are generalised notions of existential quantifiers and diagonals [22]. For this section, we fix a POM $\mathcal{M} = \langle A, \leq, \oplus, \mathbf{0} \rangle$.

**Definition 3 (Cylindrification).** *Let $V$ a set of variables. A cylindric operator $\exists$ over $\mathcal{M}$ and $V$ is a family of monotone, identity preserving functions $\exists_x : A \to A$ indexed by elements in $V$ such that for all $a, b \in A$ and $x, y \in V$*

*1. $a \leq \exists_x a$*
*2. $\exists_x(a \oplus \exists_x b) = \exists_x a \oplus \exists_x b$*
*3. $\exists_x \exists_y a = \exists_y \exists_x a$*

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

Preserving identities means that $\exists_x \mathbf{0} = \mathbf{0}$. Combined with item 2, it implies idempotency of $\exists$, i.e., $\exists_x \exists_x a = \exists_x a$, which implies $x \notin \mathrm{supp}(\exists_x a)$. Since $\exists$ is commutative, we denote $\exists_{x_1} \cdots \exists_{x_n} a$ as $\exists_X a$, for $X = \{x_1, \ldots, x_n\} \subseteq V$.

We now fix a set of variables $V$ and a cylindric operator $\exists$ over $\mathcal{M}$ and $V$.

**Definition 4 (Diagonalisation).** *A diagonal operator $\delta$ for $\exists$ is a commutative function $\delta : V \times V \to A$ such that for all $a \in A$ and $x, y, z \in V$*

*1. $\delta_{x,x} = \mathbf{0}$*
*2. $\delta_{x,y} = \exists_z(\delta_{x,z} \oplus \delta_{z,y})$ for $z \notin \{x, y\}$*
*3. $\delta_{x,y} \oplus \exists_x(a \oplus \delta_{x,y}) \leq a$ for $x \neq y$*

We use a subscript notation, as $\delta_{x,y}$ for $\delta(x, y)$. Axioms 1 and 2 above plus idempotency imply $\exists_x \delta_{x,y} = \mathbf{0}$, which implies (by axiom 2 and idempotency of $\exists$) that $\mathrm{supp}(\delta_{x,y}) = \{x, y\}$ for $x \neq y$. We lastly fix a diagonal operator $\delta$ for $\exists$.

**Definition 5 (Substitution).** *Let $x, y \in V$ and $a \in A$. The substitution $a[^y/_x]$ is defined as $a$ if $x = y$ and as $\exists_x(\delta_{x,y} \oplus a)$ otherwise.*

Substitution behaves correctly with respect to $\exists$.

**Lemma 1.** *For all $x, y, w \in V$ and $a \in A$, we have*

*– $(\exists_x a)[^y/_x] = \exists_x a$;*
*– $\exists_x a = \exists_y(a[^y/_x])$, if $y \notin \mathrm{supp}(a)$;*
*– $(\exists_w a)[^y/_x] = \exists_w(a[^y/_x])$, if $w \notin \{x, y\}$.*

*Proof.* The proofs are immediate. Consider for instance the most difficult item 3. If $x = y$ the proof is over. Now, since $w \notin \{x, y\}$ we have that $w \notin \mathrm{supp}(\delta_{x,y})$, so that $\exists_w(a[^y/_x]) = \exists_w \exists_x(\delta_{x,y} \oplus a) = \exists_x \exists_w(\delta_{x,y} \oplus a) = \exists_x(\delta_{x,y} \oplus \exists_w a) = (\exists_w a)[^y/_x]$.

Finally, we can now rephrase some additional laws that hold for the crisp case (see e.g. [7, p. 140]).

**Lemma 2.** *For all $x, y \in V$ and $a \in A$, we have*

1. $(a[^y/_x])[^x/_y] = a$, *if $y \notin \mathrm{supp}(a)$;*
2. $a[^y/_x] \oplus b[^y/_x] = (a \oplus b)[^y/_x]$;
3. $x \notin \mathrm{supp}(a[^y/_x])$, *if $x \neq y$.*

*Proof.* Consider the most difficult item 2. By definition $a[^y/_x] \oplus b[^y/_x] = \exists_x(\delta_{x,y} \oplus a) \oplus \exists_x(\delta_{x,y} \oplus b)$, which in turn coincides with $\exists_x(\delta_{x,y} \oplus a \oplus \exists_x (\delta_{x,y} \oplus b))$ by axiom 2 of $\exists$, and by axiom 3 of $\delta$ and its idempotency we have that $\exists_x(\delta_{x,y} \oplus a \oplus \exists_x(\delta_{x,y} \oplus b)) \leq \exists_x(\delta_{x,y} \oplus a \oplus b) = (a \oplus b)[^y/_x]$. The vice versa holds by the monotonicity of $\exists$, so that $\exists_x(\delta_{x,y} \oplus b) \geq \delta_{x,y} \oplus b$. Item 1 has a similar proof, while 3 is immediate.

## 3  A Key Example: Soft Constraints

Previously, we mentioned as typical examples of distributive CLMs the Fuzzy semiring $\langle [0,1], \leq, \min, 1 \rangle$ of the $[0,1]$ interval of real numbers with the usual order and multiplication as the monoidal operator, and the Tropical semiring $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +, 0 \rangle$ of non-negative reals plus $\infty$ with the inverse order and addition. In this section, we give a pivotal example of a CLM that is a cylindric algebra, introducing the notion of soft constraint (following, yet generalising [12]).

**Definition 6 (Soft constraints).** *Let $V$ be a set of variables, $\mathcal{D}$ a data domain and $\mathcal{M} = \langle A, \leq, \oplus, \mathbf{0} \rangle$ a CLM. A (soft) constraint $c : (V \to \mathcal{D}) \to A$ is a function associating a value in $A$ to every variable assignment $\eta : V \to \mathcal{D}$.*

We define $C$ as the set of constraints that can be built starting from chosen $\mathcal{M}$, $V$, and $\mathcal{D}$. The application of a constraint function $c : (V \to \mathcal{D}) \to A$ to a variable assignment $\eta : V \to \mathcal{D}$ is denoted $c\eta$.

Although a constraint involves all the variables in $V$, it may depend on the assignment of a finite subset of them, called its support (cf. Definition 3). For instance, a binary constraint $c$ with $\mathrm{supp}(c) = \{x, y\}$ is a function $c : (V \to \mathcal{D}) \to A$ which depends only on the assignment of variables $\{x, y\} \subseteq V$, meaning that two assignments $\eta_1, \eta_2 : V \to \mathcal{D}$ differing only for the image of variables $z \notin \{x, y\}$ coincide (i.e., $c\eta_1 = c\eta_2$). The support generalises the classical notion of scope of a constraint.[2] We often refer to a constraint with support $X$ as $c_X$.

The set of constraints forms a CLM, with the structure lifted from $\mathcal{M}$.

**Lemma 3 (CLM of constraints).** *The set of constraints $C$ (over $\mathcal{M}$, $V$, and $\mathcal{D}$) is endowed with a relation $\leq$, operation $\oplus$, and constant $\mathbf{0}$, such that*

- $c_1 \leq c_2$ *if $c_1\eta \leq c_2\eta$ for all $\eta : V \to \mathcal{D}$*
- $(c_1 \oplus c_2)\eta = c_1\eta \oplus c_2\eta$
- $\mathbf{0}\eta = \mathbf{0}$

*is a complete lattice monoid.*

---

[2] For a first-order constraint $\phi$, the support $\mathrm{supp}(\phi)$ is contained in the set of free variables $\mathrm{free}(\phi)$. For example, $\mathrm{supp}(x = x) = \emptyset \subseteq \{x\} = \mathrm{free}(x = x)$.

We denote the CLM $\langle C, \leq, \oplus, \mathbf{0} \rangle$ of constraints by $\mathcal{C}$. Combining two constraints by the $\oplus$ operator means building a new constraint whose support contains at most the variables of the original ones. Such constraint associates with each tuple of domain values for such variables the value obtained by multiplying those associated by the original constraints to the appropriate sub-tuples. The identity is the constant function mapping all $\eta$ to $\mathbf{0}$.

*Example 1 (A simple CLM).* Let us consider $\mathcal{S}$ as the CLM of non-negative reals, and as $\mathcal{D}$ any subset of such reals. A linear polynomial with variables in $V$ and non-negative reals as coefficients such as $ux + vy + z$ can be interpreted as the soft constraint associating with a function $\eta : V \rightarrow \mathcal{D}$ the real obtained as $(u \times \eta(x)) + (v \times \eta(y)) + z$. Clearly, the composition of such constraints is precisely the addition of polynomials. Instead, the ordering might not be the one induced by the coefficients, due to the presence of constants. For example, let us consider the polynomials $2x + 1$ and $x + 5$ and let us assume $\mathcal{D} = \{1, 2, 3\}$: it holds that $(2x + 1) \oplus (x + 5) = (3x + 6)$ and $2x + 1 \leq x + 5$.

Similarly for residuation, which is just bounded subtraction of coefficients. Since $2x + 1 \leq x + 5$, by construction $(2x + 1) \ominus (x + 5)$ is the bottom constraint, which can be represented by the polynomial 0. Instead, $(x + 5) \ominus (2x + 1)$ could be synthetically described as $-x + 4$, even if the latter falls outside of the polynomials we considered since it has a negative coefficient. In general terms, also such polynomials might be allowed: it would suffice to assume that if the result of the evaluation of the polynomial is a negative real, then it is put to 0.

If $\mathcal{D}$ is not the singleton, then the support of a polynomial is precisely the set of variables occurring in it.

The CLM of constraints enjoys the cylindric properties, as shown by the result below (for cylindric operators and diagonals in the idempotent case, see [12]).

**Lemma 4 (Cylindric algebra of constraints).** *The CLM of constraints $\mathcal{C}$ endowed with cylindric operators $\exists_x$ and diagonal elements $\delta_{x,y}$, such that*

- $(\exists_x c)\eta = \bigvee_{d \in \mathcal{D}} c\eta[x := d]$, *for all $c \in C, x \in V$*
- $\delta_{x,y}\eta = \begin{cases} \mathbf{0} & \text{if } \eta(x) = \eta(y) \\ \perp & \text{otherwise} \end{cases}$ *, for all $x, y \in V$*

*is a cylindric algebra.*

Differently from the tradition in soft constraint literature, we allow the data domain $\mathcal{D}$ to be infinite. Hence, $\exists_x$ may need to compute the least upper bound of an infinite set of soft constraints. However, Lemma 3 shows that $\mathcal{C}$ is a CLM, which guarantees the existence of such an upper bound. These observations motivate why we view the set of constraints as a CLM rather than just a POM.

Hiding means removing variables from supports: $\text{supp}(\exists_x c) \subseteq \text{supp}(c) \setminus \{x\}$.[3] Finally, the diagonal element $\delta_{x,y}$ has support $\{x, y\}$ for $x \neq y$, and $\emptyset$ for $\delta_{x,x}$.

---

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

*Example 2 (Continued... ).* Let us consider again the situation of Example 1. Polynomials can also be equipped with a cylindric operator, so that e.g. $\exists_x(2x + 1) = \bigvee_{d \in \mathcal{D}} 2d + 1 = 3$, i.e., the supremum obtained for the evaluation of the polynomial with respect to the elements in $\mathcal{D}$. The diagonal operator $\delta_{x,y}$ is a kind of matching $[x = y]$, since $[x = y]\eta$ is either 0 or $\infty$ depending if $\eta(x) = \eta(y)$ or $\eta(x) \neq \eta(y)$, respectively. A proper treatment would anyhow require to extend the syntax of polynomials by including suitable constants.

## 4    Soft Constraint Automata

Constraint Automata (CA) have bee introduced in [8] as a formalism to describe the behaviour and possible data flow in coordination models (such as Reo language [8]); they can be considered as acceptors of *Timed Data Streams* [1,8]. The proposal has been recently enriched by adding an explicit notion of memory [18]. In this section we rephrase most definitions presented in the weighted extension of CAs, namely *Soft Constraint Automata* (SCA) [6], and we further extend the framework by relying on CLMs instead of c-semirings as in [6] and by including memory, thus obtaining Soft Constraint Automata *with Memory* (SCAM).

### 4.1    Weighted Data Streams

As a first step, we recall and rephrase the definition of *Weighted Data Streams* (WDS), which is also given in [6].[4]

In this section we fix a data domain $\mathcal{D}$ as well as a CLM $\mathcal{M} = \langle A, \leq, \oplus, \mathbf{0} \rangle$.

**Definition 7 (Weighted data streams).** *A weighted data stream (WDS) over $\mathcal{D}$ and $\mathcal{M}$ is a partial function $\phi : \mathbb{R}_+ \rightharpoonup (A \setminus \{\bot\}) \times \mathcal{D}$ whose domain $\mathrm{dom}(\phi)$ is closed and discrete. We write WDS for the set of all weighted data streams.*

Intuitively, a WDS $\phi : \mathbb{R}_+ \rightharpoonup (A \setminus \{\bot\}) \times \mathcal{D}$ records the time stamps $\mathrm{dom}(\phi) = \{\phi_0, \phi_1, \ldots\}$, such that $i < j$ implies $\phi_i < \phi_j$, at which data is exchanged. Indeed, discreteness (each element is isolated) ensures that $\mathrm{dom}(\phi)$ is countable (hence, it has the cardinality of a subset of natural numbers). Since it is also closed, the domain contains no unrealistic situations like $\{1/n \mid n \in \mathbb{N}\}$, since its limit would not be discrete. The values $\phi(\phi_i) = (a_i, d_i)$, for $i \geq 0$ and $\phi_i \in \mathrm{dom}(\phi)$, consists of the observed data $d_i$ and its preference $a_i \neq \bot$.

Our current definition of WDSs slightly differs from the original definition in [6]. The main distinction is that we allow finite WDSs, i.e., the domain $\mathrm{dom}(\phi)$ may be finite. In case of an empty domain $\mathrm{dom}(\phi) = \emptyset$, WDS $\phi$ admits no observable behaviour at all. This generalisation is especially useful for the hiding operator on SCAM, as proposed in Sect. 4.6.

The second difference with the original definition in [6] is that our WDSs are partial functions, rather than triples of streams. However, our assumptions on the domain imply that these representations are similar.

---

[4] As noted in [6], these streams do not imply time constraints, and thus our (soft) CA are not "timed" [8], so that we dropped the adjective altogether.

The behaviour of a system is often described in terms of tuples of WDSs, each one corresponding to a data passing through a given port; put simply, given a finite set of port names $\mathcal{N}$, the behaviour is described by a function $\mathcal{N} \to WDS$.

## 4.2   Soft Constraint Automata with Memory

In a CA, a transition label is a pair $\langle N, g \rangle$ consisting of a synchronisation constraint $N$ and a data constraint $g$. The synchronisation constraint is a finite set of names that consists exactly of those input/output ports through which data is exchanged during the current transition. The data constraint is a boolean formula that guards the data exchanged at the ports in $N$. In the soft framework, the overall structure is similar, even if the guard is now a soft constraint that evaluates to an element in $\mathcal{M}$, instead of a boolean value. Furthermore, we extend our data constraints with memory variables. Using these variables, a guard can require a property about the current and next state of the data in memory.

Besides a data domain $\mathcal{D}$ and a finite set of port names $\mathcal{N}$, we fix a finite set $\mathcal{X}$ of memory cells and a CLM $\mathcal{M} = \langle A, \leq, \oplus, \mathbf{0} \rangle$. Furthermore, we consider the set $\mathcal{X}^v = {}^{\bullet}\mathcal{X} \cup \mathcal{X}^{\bullet}$ of memory variables: they are obtained by tagging the memory cells. To avoid conflicting names, we assume that $\mathcal{N} \cap \mathcal{X}^v = \emptyset$.

**Definition 8 (Soft constraints with memory).** *A soft constraint with memory over $\mathcal{N}$, $\mathcal{X}$, $\mathcal{D}$, and $\mathcal{M}$ is a soft constraint $c : (\mathcal{N} \cup \mathcal{X}^v \to \mathcal{D}) \to A$.*

We denote by $\mathcal{C}_{\mathcal{X}}$ the CML of soft constraints with memory. Informally, a soft constraint with memory is a function that returns a preference value $a \in A$ given an assignment for a subset $N$ of names in $\mathcal{N}$ and a subset $X$ of variables in $\mathcal{X}^v$ that occur in its support.

Note that, by using the *Boolean* semiring we model the "crisp" data-constraints presented in the definition of CA [8]. Therefore, CA are subsumed by Definition 9. Note also that weighted automata have already been defined in the literature [13]; in SCA, weights are determined by a constraint function instead.

**Definition 9 (Soft constraint automata with memory).** *A Soft Constraint Automaton with Memory (SCAM) over $\mathcal{D}$ and $\mathcal{M}$ is a tuple $\langle \mathcal{Q}, \mathcal{X}, \mathcal{N}, \longrightarrow, \mathcal{Q}_0 \rangle$ such that $\mathcal{Q}$ is a finite set of states, $\mathcal{X}$ a finite set of memory cells, $\mathcal{N}$ a finite set of port names, $\longrightarrow \subseteq \mathcal{Q} \times 2^{\mathcal{N}} \times \mathcal{C}_{\mathcal{X}} \times \mathcal{Q}$ a finite set of transitions, and $\mathcal{Q}_0 \subseteq \mathcal{Q}$ a set of initial states, such that $(q, N, c, p) \in \longrightarrow$ implies $\operatorname{supp}(c) \subseteq N \cup \mathcal{X}^v$.*

We usually write $q \xrightarrow{N,c} p$ instead of $(q, N, c, p) \in \longrightarrow$ and we call $N$ the synchronisation constraint and $c$ the guard of the transition, respectively. We say a transition is *invisible* whenever $N = \emptyset$.

The intuitive meaning of a SCAM $\mathcal{T}$ as an operational model for service queries is similar to the interpretation of labelled transition systems as models for reactive systems. The states represent the configurations of a service. The transitions represent the possible one-step behaviour, where the meaning of $q \xrightarrow{N,c} p$ is that, in configuration $q$, the ports in $n \in N$ have the possibility of

performing I/O operations that satisfy the soft guard $c$, which now also includes requirements on memory cells, and that leads from configuration $q$ to $p$, while the other ports in $\mathcal{N}\backslash N$ perform no I/O operation. Each assignment to port names in $N$ represents the data exchanged by the I/O operations through these ports, while the assignments to variables in ${}^{\bullet}X$ and $X^{\bullet}$ represent the data in memory cells before and after the transition.

*Example 3 (Buying and selling).* In Fig. 1, we show a (deterministic) SCAM, where the set of names $\mathcal{N}$ is $\{b, s\}$, the set of variables $\mathcal{X}^v$ is $\{{}^{\bullet}a, a^{\bullet}, {}^{\bullet}l, l^{\bullet}\}$, and the data domain consists of all integers. The constraints $c_b$ and $c_s$ describe the preferences on the operation of buying and selling: if the cost $b$ of buying an item is below a threshold with respect to the value in the account ${}^{\bullet}a$ (for the sake of simplicity, the account remains positive, i.e., ${}^{\bullet}a - b > 0$), then the item is bought and the account is decreased ($a^{\bullet} = {}^{\bullet}a - b$ and $l^{\bullet} = b$). Likewise, if the price $s$ received in selling an item is above the paid price (${}^{\bullet}l < s$), then it is accepted and the account is incremented ($a^{\bullet} = {}^{\bullet}a + s$).



$$\langle\{b\}, c_b\rangle$$

$$\text{start} \rightarrow \boxed{q_0} \quad \boxed{q_1}$$

$$\langle\{s\}, c_s\rangle$$
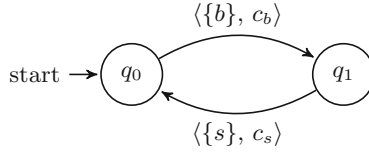
**Fig. 1.** An automaton with memory.

The formal definition of constraints $c_b$ and $c_s$ is given in Eqs. 1 and 2.

$$c_{\{b\}}\eta = \begin{cases} -\eta(b) \ \textit{if} \ 0 < \eta(l^{\bullet}) = \eta(b) = \eta({}^{\bullet}a) - \eta(a^{\bullet}) < \eta({}^{\bullet}a) \\ \bot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textit{otherwise} \end{cases} \tag{1}$$

$$c_{\{s\}}\eta = \begin{cases} \eta(s) \ \textit{if} \ \eta({}^{\bullet}l) < \eta(s) = \eta(a^{\bullet}) - \eta({}^{\bullet}a) \\ \bot \qquad\qquad\qquad\qquad\quad \textit{otherwise} \end{cases} \tag{2}$$

For example, let us assume that for the first two transitions (from $q_0$ to $q_1$ and from $q_1$ to $q_0$, respectively) we have $\eta(b) = 2$ and then $\eta(s) = 3$. Also, let us assume that we have as initial account $a = 6$ (i.e., ${}^{\bullet}a = 6$). Now, the constraint $c_{\{b\}}\eta$ associated to the first transition has value $-2$, since the money is spent, and the values associated to the memory cells $l$ and $a$ after the transition (i.e., to the variables $l^{\bullet}$ and $a^{\bullet}$) are 2 and 4, respectively. The constraint $c_{\{s\}}\eta$ associated to the second transition has value 3, since the money is earned (and it is more than what the item was paid), and the value associated to the memory cell $a$ after the transition (i.e., to the variable $a^{\bullet}$) is 7, while the value associated to the memory cell $l$ (i.e., to the variable $l^{\bullet}$) is irrelevant, hence it can be any value.

### 4.3   The Language of SCAM

Let $\mathcal{T}$ be a SCAM and $\Omega_\mathcal{X} : \mathcal{X} \to \mathcal{D}$ the set of assignments over its memory cells. The configurations of $\mathcal{T}$ are pairs $\langle q, \omega \rangle \in \mathcal{Q} \times \Omega_\mathcal{X}$, which are initial whenever $q \in \mathcal{Q}_0$. In other words, we initialise each memory cell to a random datum.

The accepted language of a SCAM $\mathcal{T}$ at a given configuration $s$ is a relation $\mathcal{L}(\mathcal{T}, s) \subseteq WDS^\mathcal{N}$ over weighted data streams on ports from $\mathcal{N}$. An accepted word consists of a $\mathcal{N}$-tuple of weighted data streams (i.e., a map $\mathcal{N} \to WDS$). As usual, the language of an automaton is given by the union of all the languages accepted by its initial states, i.e., $\mathcal{L}(\mathcal{T}) = \bigcup_{s \in \mathcal{Q}_0 \times \Omega_\mathcal{X}} \mathcal{L}(\mathcal{T}, s)$.

Consider the automaton $\mathcal{T}$ from Example 3, with ports $\mathcal{N} = \{b, s\}$ and starting state $q_0$. The accepted language of $\mathcal{T}$ is formed by the runs of alternate assignments for $\{b\}$ and $\{s\}$, which guarantees that a balance $a$ is always positive.

Recall that $\phi_0 \in \mathbb{R}_+$ is the minimum of the domain $\mathrm{dom}(\phi)$, for every WDS $\phi : \mathbb{R}_+ \rightharpoonup A \times \mathcal{D}$ with non-empty domain.

**Definition 10 (Accepted runs).** *Let $\mathcal{T} = \langle \mathcal{Q}, \mathcal{X}, \mathcal{N}, \longrightarrow, \mathcal{Q}_0 \rangle$ be a SCAM. The accepted language of $\mathcal{T}$ is the largest map $\mathcal{L}(\mathcal{T}, -) : \mathcal{Q} \times \Omega_\mathcal{X} \to 2^{(WDS)^\mathcal{N}}$, such that if $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T}, \langle q, \omega_0 \rangle)$ and $t_0 = \min\{\phi_0^x \mid x \in \mathcal{N}, \mathrm{dom}(\phi^x) \neq \emptyset\}$ exist, there exist a transition $q \xrightarrow{N, c} p$ and an assignment $\eta : \mathcal{N} \cup \mathcal{X}^v \to \mathcal{D}$ with*

- *$c\eta \neq \bot$,*
- *$N = \emptyset$ or $N = \{x \in \mathcal{N} \mid \phi_0^x = t_0\}$,*
- *$\phi^x(t_0) = \langle c\eta, \eta(x) \rangle$, for all $x \in N$,*
- *$\eta(^\bullet x) = \omega_0(x)$, for all $x \in \mathcal{X}$,*
- *$(\phi^x|_{\mathbb{R}_+ \setminus \{t_0\}})_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T}, \langle p, \omega_1 \rangle)$, with $\omega_1(x) = \eta(x^\bullet)$, for all $x \in \mathcal{X}$,*

*where $\phi^x|_{\mathbb{R}_+ \setminus \{t_0\}}$ is the restriction of $\phi^x$ to $\mathbb{R}_+$ without $t_0$. The set $\mathcal{L}(\mathcal{T})$ of accepted runs is the union of the acceptable runs from $\mathcal{Q}_0 \times \Omega_\mathcal{X}$.*

Observe that the expression $\phi^x(t_0)$ is well-defined, because $x \in N$ implies $N \neq \emptyset$ and $t_0 = \phi_0^x \in \mathrm{dom}(\phi^x)$. Also note that, since $\mathrm{supp}(c) \subseteq N \times \mathcal{X}^v$, the value of $\eta(x)$, for $x \in \mathcal{N} \setminus N$, is irrelevant.

If we never observe any data at any port (i.e., $\mathrm{dom}(\phi^x) = \emptyset$, for all $x \in \mathcal{N}$), then the minimum $t_0 = \min\{\phi_0^x \mid x \in \mathcal{N}, \mathrm{dom}(\phi^x) \neq \emptyset\}$ does not exist, and the condition in Definition 10 is vacuously true.

*Example 4 (The language of business).* Going back to Example 3, the language recognised by state $q_0$ with respect to the memory cell assignment $[a = p, l = 0]$ is the possibly infinite sequence of weights and port name assignments $\langle -b_1, b = b_1 \rangle; \langle s_1 - b_1, s = s_1 \rangle; \ldots$ such that $0 < b_i \leq s_i$ and additionally in any (even) prefix of such a sequence the sum of all the costs and all the sales is always positive. Indeed, it is given by the sum of all constraints! The timing of these operations is instead irrelevant and it is omitted.

### 4.4   Stateless SCAM

States and memory cells of SCAM essentially serve the same purpose. The following theorem shows that we can eliminate all states from any SCAM, at the cost of new memory cells.

**Theorem 1.** *Let $\mathcal{T}$ be a SCAM over a domain $\mathcal{D}$ that is not a singleton. Then, it is language equivalent to a SCAM with only a single state.*

*Proof.* Let $\mathcal{T} = (\mathcal{Q}, \mathcal{X}, \mathcal{N}, \longrightarrow, \mathcal{Q}_0)$ be a SCAM over a data domain $\mathcal{D}$ and a CLM $\mathcal{M}$ such that $\mathcal{D}$ is not a singleton. We find an injective encoding $e : Q \to \mathcal{D}^n$, for some integer $n \geq \log_{|\mathcal{D}|}(|Q|)$. Write $e(q) = (e_i(q))_{i=1}^n$, for all $q \in Q$. For every variable $x \in \mathcal{X}^v$ and datum $d \in \mathcal{D}$, define the (soft) constraint $x = d$ by $(x = d)\eta = \mathbf{0}$, if $\eta(x) = e_i(q)$, and $(x = d)\eta = \bot$, otherwise. Let $z_1, \ldots, z_n \notin \mathcal{N} \cup \mathcal{X}$ be fresh names. For every $\tau = (q, N, c, p) \in \longrightarrow$, define $N_\tau = N$ and

$$c_\tau = c \oplus \bigoplus_{i=1}^n {}^\bullet z_i = e_i(q) \oplus z_i^\bullet = e_i(p).$$

Consider $\mathcal{T}' = (\{q_0\}, \mathcal{X} \cup \{z_1, \ldots, z_n\}, \mathcal{N}, \{(q_0, N_\tau, c_\tau, q_0) \mid \tau \in \longrightarrow\}, \{q_0\})$. We show that $\mathcal{T}'$ is language equivalent to $\mathcal{T}$. Suppose that $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T})$. Then, we find some transition $(q, N, c, p) \in \longrightarrow$ and assignment $\eta : \mathcal{N} \cup \mathcal{X}^v \to \mathcal{D}$ that satisfy the conditions in Definition 10. Since the $z_i$'s are fresh, we find an extension $\eta'$ of $\eta$ to $\bigcup_{i=1}^n \{{}^\bullet z_i, z_i^\bullet\}$ that satisfies $c_\tau \eta' \neq \bot$. Hence, transition $(q_0, N_\tau, c_\tau, q_0)$ and assignment $\eta'$ witness that $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T}')$.

On the other hand, suppose that $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T}')$. Then, we find some transition $(q_0, N_\tau, c_\tau, q_0)$, with $\tau \in \longrightarrow$, and an assignment $\eta' : \mathcal{N} \cup (\mathcal{X} \cup \{z\})^v \to \mathcal{D}$ that satisfy the conditions in Definition 10. Then, $\tau$ and the restriction $\eta$ of $\eta'$ to $\mathcal{N} \cup \mathcal{X}^v$ witness that $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T})$. We conclude that $\mathcal{L}(\mathcal{T}) = \mathcal{L}(\mathcal{T}')$. □

The result of Theorem 1 can be used as a first step towards an equivalence between SCAM and soft constraints with memory (over a data domain that includes a special datum $*$ that denotes absence of data). Moreover, by using $*$, it seems possible to encode the synchronisation constraint $N$ of each transition in the soft constraint by adding $x \neq *$, for $x \in N$, and $x = *$, otherwise. In combination with Theorem 1, such construction would show a correspondence between SCAM and soft constraints with memory. Since the representation of SCAM as soft constraints with memory is much more flexible, this alternative representation can help us to prevent relentless state space explosions. We leave the details of such correspondence as future work.

### 4.5   SCAM Composition

We now introduce the product of automata, extending [6, Definition 5].

**Definition 11 (Soft join).** *Let $\mathcal{T}_i = (\mathcal{Q}_i, \mathcal{X}_i, \mathcal{N}_i, \to_i, \mathcal{Q}_{0i})$, for $i \in \{0, 1\}$, be two SCAM over $\mathcal{D}$ and $\mathcal{M}$, with $(\mathcal{N}_0 \cup \mathcal{N}_1) \cap (\mathcal{X}_0^v \cup \mathcal{X}_1^v) = \emptyset$. Then, their soft*

*product $\mathcal{T}_0 \bowtie \mathcal{T}_1$ is the tuple $\langle \mathcal{Q}_0 \times \mathcal{Q}_1, \mathcal{X}_0 \cup \mathcal{X}_1, \mathcal{N}_0 \cup \mathcal{N}_1, \longrightarrow, \mathcal{Q}_{00} \times \mathcal{Q}_{01} \rangle$ where $\longrightarrow$ is the smallest relation that satisfies the rule*

$$\frac{q_0 \xrightarrow{N_0, c_0}_0 p_0, \quad q_1 \xrightarrow{N_1, c_1}_1 p_1, \quad N_0 \cap \mathcal{N}_1 = N_1 \cap \mathcal{N}_0}{\langle q_0, q_1 \rangle \xrightarrow{N_1 \cup N_2, c_1 \oplus c_2} \langle p_0, p_1 \rangle}$$

The rule applies when there is a transition in each automaton such that they can fire together. This happens only if the two local transitions agree on the subset of shared ports that *fire*. The transition in the resulting automaton is labelled with the union of the name sets on both transitions, and the constraint is the conjunction of the constraints of the two transitions.

Note that the product automaton can include asynchronous executions: it suffices that the SCAM are *reflexive*, i.e., for any state $q$ there is a transition $q \xrightarrow{\emptyset, \mathbf{0}} q$.

We can express the composition of SCAM in Definition 11 in terms of a simple composition operator on languages. Let $\mathcal{L}_0$ and $\mathcal{L}_1$ be two languages over the sets of ports $\mathcal{N}_0$ and $\mathcal{N}_1$, respectively. The product language $\mathcal{L}_0 \bowtie \mathcal{L}_1$ consists of those tuples $(\phi^x)_{x \in \mathcal{N}_0 \cup \mathcal{N}_1}$ of WDSs, such that $(\phi^x)_{x \in \mathcal{N}_i} \in \mathcal{L}_i$, for all $i \in \{0, 1\}$. In particular, $\mathcal{N}_0 = \mathcal{N}_1$ implies $\mathcal{L}_0 \bowtie \mathcal{L}_1 = \mathcal{L}_0 \cap \mathcal{L}_1$.

**Lemma 5 (Correctness of soft join).** *Let $\mathcal{T}_0$ and $\mathcal{T}_1$ be two SCAM that do not share memory cells. Then, $\mathcal{L}(\mathcal{T}_0 \bowtie \mathcal{T}_1) = \mathcal{L}(\mathcal{T}_0) \bowtie \mathcal{L}(\mathcal{T}_1)$.*

*Proof.* Suppose that $(\phi^x)_{x \in \mathcal{N}_0 \cup \mathcal{N}_1} \in \mathcal{L}(\mathcal{T}_0 \bowtie \mathcal{T}_1)$. We show that $(\phi^x)_{x \in \mathcal{N}_i} \in \mathcal{L}(\mathcal{T}_i)$, for all $i \in \{0, 1\}$. Let $i \in \{0, 1\}$ be arbitrary. By Definition 10, we find some transition $q \xrightarrow{N, c} p$ and an assignment $\eta : \mathcal{N}_0 \cup \mathcal{N}_1 \cup \mathcal{X}_0^v \cup \mathcal{X}_1^v \to A$ that satisfy the conditions in Definition 10. By Definition 11, we find a local transition $q_i \xrightarrow{N_i, c_i} p_i$ in $\mathcal{T}_i$, and by restriction of $\eta$, we find a local assignment $\eta_i : \mathcal{N}_i \cup \mathcal{X}_i^v \to A$. By construction, this transition and assignment witness the inclusion $(\phi^x)_{x \in \mathcal{N}_i} \in \mathcal{L}(\mathcal{T}_i)$ according to the conditions in Definition 10. By definition of join, we have $\mathcal{L}(\mathcal{T}_0 \bowtie \mathcal{T}_1) \subseteq \mathcal{L}(\mathcal{T}_0) \bowtie \mathcal{L}(\mathcal{T}_1)$.

Suppose that $(\phi^x)_{x \in \mathcal{N}_0 \cup \mathcal{N}_1} \in \mathcal{L}(\mathcal{T}_0) \bowtie \mathcal{L}(\mathcal{T}_1)$. We show that $(\phi^x)_{x \in \mathcal{N}_0 \cup \mathcal{N}_1} \in \mathcal{L}(\mathcal{T}_0 \bowtie \mathcal{T}_1)$. By definition, we have $(\phi^x)_{x \in \mathcal{N}_i} \in \mathcal{L}(\mathcal{T}_i)$, for all $i \in \{0, 1\}$. By Definition 11, we find, for $i \in \{0, 1\}$, a transition $q_i \xrightarrow{N_i, c_i} p_i$ and an assignment $\eta_i : \mathcal{N}_i \cup \mathcal{X}_i^v \to A$ that satisfy the conditions in Definition 10. By construction, we have $N_0 \cap \mathcal{N}_1 = N_1 \cap \mathcal{N}_0$, and we find with Definition 11 a global transition $q \xrightarrow{N, c} p$ in $\mathcal{T}_0 \bowtie \mathcal{T}_1$. Since $\mathcal{T}_0$ and $\mathcal{T}_1$ do not share memory, we have $\mathcal{X}_0^v \cap \mathcal{X}_1^v = \emptyset$. Therefore, $\eta_0 \cup \eta_1$ is a well-defined assignment that, together with $q \xrightarrow{N, c} p$, witnesses that $(\phi^x)_{x \in \mathcal{N}_0 \cup \mathcal{N}_1} \in \mathcal{L}(\mathcal{T}_0 \bowtie \mathcal{T}_1)$. We conclude that $\mathcal{L}(\mathcal{T}_0 \bowtie \mathcal{T}_1) = \mathcal{L}(\mathcal{T}_0) \bowtie \mathcal{L}(\mathcal{T}_1)$. □

### 4.6 SCAM Hiding

The hiding operator [8] abstracts the details of the internal communication in a CA. In SCA [6, Definition 6], the hiding operator $\exists_O \mathcal{T}$ removes from the

transitions all the information about the names in $O \subseteq \mathcal{N}$, including those in the (support of the) constraints. The definition smoothly extends over SCAM: in fact, since we allow invisible transitions, our definition is much more compact.

**Definition 12 (Soft hiding).** *Let $\mathcal{T} = \langle \mathcal{Q}, \mathcal{X}, \mathcal{N}, \longrightarrow, \mathcal{Q}_0 \rangle$ be a SCAM and $O \subseteq \mathcal{N}$ a set of port names. Then, $\exists_O \mathcal{T}$ is the SCAM $\langle \mathcal{Q}, \mathcal{X}, \mathcal{N} \setminus O \longrightarrow_*, \mathcal{Q}_0 \rangle$ where $\longrightarrow_*$ is defined by $q \xrightarrow{N \setminus O, \exists_O c}_* p$ if $q \xrightarrow{N,c} p$.*

Similarly to the correctness of join, we express the hiding operator for SCAM in terms of a simple operation on languages. Let $\mathcal{L}$ be a language over a set of ports $\mathcal{N}$, and let $O \subseteq \mathcal{N}$ be a set of ports. Then, $\exists_O \mathcal{L}$ consists exactly of the restriction $(\phi^x)_{x \in \mathcal{N} \setminus O}$ of a given tuple $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}$.

If we require that the domain $\mathrm{dom}(\phi)$ of a WDS $\phi$ is infinite, then we can only hide ports that necessarily fire infinitely often in every (infinite) run of the SCAM. This condition is part of the correctness of hiding for SCA [6, Definition 6]. Since we allow finite WDS, correctness of SCAM amounts to the following result.

**Lemma 6 (Correctness of soft hiding).** *Let $\mathcal{T}$ be a SCAM and $O$ a set of its ports. Then, $\mathcal{L}(\exists_O \mathcal{T}) = \exists_O \mathcal{L}(\mathcal{T})$.*

*Proof.* Suppose that $(\phi^x)_{x \in \mathcal{N} \setminus O} \in \mathcal{L}(\exists_O \mathcal{T})$. We will show by coinduction that $(\phi^x)_{x \in \mathcal{N} \setminus O} \in \exists_O \mathcal{L}(\mathcal{T})$. By Definition 10, we find some transition $q \xrightarrow{N', c'} p$ in $\exists_O \mathcal{T}$. By Definition 12, we find some transition $q \xrightarrow{N,c} p$, with $N' = N \setminus O$ and $c' = \exists_O c$. We construct a WDS $\phi^x$, for $x \in O$, such that $(\phi^x)_{x \in \mathcal{N}}$ satisfies the conditions of Definition 10, for $\mathcal{T}$. If $x \in N \cap O$, we define $\phi^x = [t_0 \mapsto \langle c\eta, \eta(d) \rangle]$, with $t_0 := \min\{\phi_0^p \mid p \in \mathcal{N} \setminus O, \mathrm{dom}(\phi^p) \neq \emptyset\}$ and $\eta : \mathcal{N} \cup \mathcal{X}^v \to \mathcal{D}$ is any assignment that satisfies $c$. If $x \in (\mathcal{N} \setminus N) \cap O$ and we define $\phi^x = [\,]$ as the empty map. By the coinduction hypothesis, $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T})$, and $(\phi^x)_{x \in \mathcal{N} \setminus O} \in \exists_O \mathcal{L}(\mathcal{T})$.

On the other hand, suppose that $(\phi^x)_{x \in \mathcal{N} \setminus O} \in \exists_O \mathcal{L}(\mathcal{T})$. By definition, we find some extension $(\phi^x)_{x \in \mathcal{N}} \in \mathcal{L}(\mathcal{T})$. By coinduction hypothesis and Definition 10 we find $(\phi^x)_{x \in \mathcal{N} \setminus O} \in \mathcal{L}(\exists_O \mathcal{T})$. We conclude that $\mathcal{L}(\exists_O \mathcal{T}) = \exists_O \mathcal{L}(\mathcal{T})$.    □

## 5   Related Works on Constraint Automata

The closest related work to what discussed in this paper concerns other extensions of CA, previously advanced in the ample and mature literature about Reo.

In [5,21], Arbab et al. introduce *Quantitative* CA (QCA) with the aim of describing the behaviour of connectors tied to their *Quality of Service* (QoS), e.g., a reliability measure or the shortest transmission time. Similarly to CA, the states of a QCA correspond to the internal states of the connector it models. The label on a transition consists instead of a firing set, a data constraint, and a cost that represents a QoS metric. Hence, QCA differ from *Timed* [4] and *Probabilistic* [9] CA, because these latter two classes of models describe functional aspects of connectors, while QoS represents non-functional properties.

As applications, SCA have been already used in [6,23] and [19,20,24]. Differently from previous related work, the main motivation behind SCA is to associate an action with a preference. In [6,23] the authors present a formal framework that is able to discover stateful Web Services, and to rank the results according to a similarity score expressing the affinities between the query, asked by a user, and the services in a database. Preference for the similarity between the query and each service is modelled through SCA. In the second bunch of works instead, the authors advance a framework that facilitates the construction of autonomous agents in a compositional fashion; these agents are "soft", in that their actions are associated with a preference value, and agents may or may not execute an action depending on a threshold preference. Hence, at design-time SCA can be used to reason about the behaviour of the components in an uncertain physical world, i.e., to model and verify the behaviour of cyber-physical systems.

## 6    Conclusions

We have reworked Soft Constraint Automata as originally proposed in [6,19], with the dual purpose of *(i)* extending the underlying algebraic structure in order to model both positive and negative preferences, and *(ii)* adding memory cells as originally provided for "standard" CA [18]. Therefore, the main objective has been to further generalise the notion of SCA, which could already accommodate different preference systems parametrically.

As future work, we have many interesting directions in mind. As a start, we would like to exploit the properties of soft constraints to give additional operators on SCAM, first of all an operator for port *renaming*, or for considering *deterministic* accepted runs, i.e., where memory cells that are not in the support of a constraint labelling a transition are not modified by that transition. We will consider a more flexible notion of accepted run by taking into account *weak* transitions, i.e., by considering the relation $q \overset{\emptyset,c}{\Longrightarrow} p$ obtained as the sequence $q \overset{\emptyset,c_1}{\longrightarrow} q_1 \ldots q_{n-1} \overset{\emptyset,c_n}{\longrightarrow} p$ such that $c = c_1 \oplus \ldots \oplus c_n$ and the relation $q \overset{N,c}{\Longrightarrow} p$ obtained as $q \overset{\emptyset,c_1}{\Longrightarrow} q_1 \overset{N,c_2}{\longrightarrow} p_1 \overset{\emptyset,c_3}{\Longrightarrow} p$ such that $c = c_1 \oplus c_2 \oplus c_3$. This would be pivotal in defining a proper notion of *weak bisimulation* for automata.

Finally, thinking about our result on single state automata, we would like to encode the behaviour of SCA into a *concurrent constraint programming* language [15]. Such languages provide agents with actions to *tell* (i.e., add) and *ask* (i.e., query) constraints to a centralised store of information; this store represents a *Constraint Satisfaction Problem*, and standard heuristics-based technique might be applied to find a solution to complex conditions on *filter channels* [3].

# References

1. Arbab, F., Rutten, J.J.M.M.: A coinductive calculus of component connectors. In: Wirsing, M., Pattinson, D., Hennicker, R. (eds.) WADT 2002. LNCS, vol. 2755, pp. 34–55. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40020-2_2

2. Arbab, F.: Reo: a channel-based coordination model for component composition. Math. Struct. Comput. Sci. **14**(3), 329–366 (2004)

3. Arbab, F.: Puff, the magic protocol. In: Agha, G., Danvy, O., Meseguer, J. (eds.) Formal Modeling: Actors, Open Systems, Biological Systems. LNCS, vol. 7000, pp. 169–206. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24933-4_9

4. Arbab, F., Baier, C., de Boer, F.S., Rutten, J.J.M.M.: Models and temporal logical specifications for timed component connectors. Softw. Syst. Model. **6**(1), 59–82 (2007)

5. Arbab, F., Chothia, T., Meng, S., Moon, Y.-J.: Component connectors with QoS guarantees. In: Murphy, A.L., Vitek, J. (eds.) COORDINATION 2007. LNCS, vol. 4467, pp. 286–304. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72794-1_16

6. Arbab, F., Santini, F.: Preference and similarity-based behavioral discovery of services. In: ter Beek, M.H., Lohmann, N. (eds.) WS-FM 2012. LNCS, vol. 7843, pp. 118–133. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38230-7_8

7. Aristizábal, A., Bonchi, F., Palamidessi, C., Pino, L., Valencia, F.: Deriving labels and bisimilarity for concurrent constraint programming. In: Hofmann, M. (ed.) FoSSaCS 2011. LNCS, vol. 6604, pp. 138–152. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19805-2_10

8. Baier, C., Sirjani, M., Arbab, F., Rutten, J.J.M.M.: Modeling component connectors in Reo by constraint automata. Sci. Comput. Program. **61**(2), 75–113 (2006)

9. Baier, C.: Probabilistic models for Reo connector circuits. Univers. Comput. Sci. **11**(10), 1718–1748 (2005)

10. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. J. ACM **44**(2), 201–236 (1997)

11. 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)

12. Bistarelli, S., Montanari, U., Rossi, F.: Soft concurrent constraint programming. ACM Trans. Comput. Logic **7**(3), 563–589 (2006)

13. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01492-5

14. Gadducci, F., Santini, F.: Residuation for bipolar preferences in soft constraints. Inf. Process. Lett. **118**, 69–74 (2017)

15. Gadducci, F., Santini, F., Pino, L.F., Valencia, F.D.: Observational and behavioural equivalences for soft concurrent constraint programming. Log. Algebr. Methods Program. **92**, 45–63 (2017)

16. Golan, J.: Semirings and Affine Equations over Them: Theory and Applications. Kluwer, Norwell (2003)

17. Jongmans, S.T.Q., Arbab, F.: Overview of thirty semantic formalisms for Reo. Sci. Ann. Comput. Sci. **22**(1), 201–251 (2012)

18. Jongmans, S.T.Q., Kappé, T., Arbab, F.: Constraint automata with memory cells and their composition. Sci. Comput. Program. **146**, 50–86 (2017)
19. Kappé, T., Arbab, F., Talcott, C.L.: A compositional framework for preference-aware agents. In: Kargahi, M., Trivedi, A. (eds.) V2CPS@IFM 2016. EPTCS, vol. 232, pp. 21–35 (2016)
20. Kappé, T., Arbab, F., Talcott, C.: A component-oriented framework for autonomous agents. In: Proença, J., Lumpe, M. (eds.) FACS 2017. LNCS, vol. 10487, pp. 20–38. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68034-7_2
21. Meng, S., Arbab, F.: QoS-driven service selection and composition using quantitative constraint automata. Fundamenta Informaticae **95**(1), 103–128 (2009)
22. 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)
23. Sargolzaei, M., Santini, F., Arbab, F., Afsarmanesh, H.: A tool for behaviour-based discovery of approximately matching web services. In: Hierons, R.M., Merayo, M.G., Bravetti, M. (eds.) SEFM 2013. LNCS, vol. 8137, pp. 152–166. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40561-7_11
24. Talcott, C., Nigam, V., Arbab, F., Kappé, T.: Formal specification and analysis of robust adaptive distributed cyber-physical systems. In: Bernardo, M., De Nicola, R., Hillston, J. (eds.) SFM 2016. LNCS, vol. 9700, pp. 1–35. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-34096-8_1