Working title

Abstract. Soft constraint automata extend constraint automata by associating, for each transition, a preference value from a csemiring. Preferences model concerns, and multiple concern can be involved in a single choice. We present a new framework to compose preferences of difference csemiring using a co-product. Since constraint automata have an interpretation as a boolean formula, we then investigate the interpretation of soft constraint automata as a semiring formula.

Preferences occur when you have to make a choice. Imagine you are walking on a trekking path, and you meet an intersection. At this point, two actions are possible : turn left, or turn right. Since you don't have any problem with those two actions, you can potentially do both. Suppose, in this case, that no other elements help you to determine your choice, you will chose in a non deterministic way. Now, suppose that you can read the distance from your point to your destination taking the right path or left path. Then, you will be able to chose according to the shortest path. Constraint semiring are suitable structures to define preferences and their composition.

Soft Constraint Automata (SCA) have been developed by Arbab & Santini [1] and constitute a solid ground to design and reason about systems with preferences. Instead of being restricted to crisp constraint, SCA defines soft constraints, based on c-semiring, and can choose the most preferred among a set of choices. Besides a choice operator, composition is also possible and results into new preferences. Defining global preferences by composition of local preferences is a design choice that we do not justify in this paper. Intuitively, the objective is to be able to remove an error of the composed automata by modifying preferences in the composite automata. A diagnosis procedure has been detailed by [3]. Thus, composition is at the core of the definition of Soft Constraint systems.

Until now, several composition operators have been defined (ex: lexicographic, join). The choice of the product space must be done before composition.

In this paper, we first present the definition of a new composition framework for preferences in Soft Constraint Automata, by using the co-product. Composition is decoupled from quotient to product space. Lexicographic or join quotients are defined.

Then, the definition of a new concise model to represent, in a unified way, constraints and preferences into a semiring logic. Logic is extended to any csemiring (not only boolean csemiring).

1 Definition

1.1 Constraint Automata

Constraint Automata (CA) were introduced in [4] and define a compositional framework for the design of concurrent systems. CA are state transition system where transitions are labeled with constraints. We first present the language that defines the constraints, and then define the syntax and semantic of Constraint Automata.

Definition 1. *(Language of constraint) A term is defined as:*

$$t := \quad v \quad | \quad f(t_1, ..., t_n) \quad | \quad *$$

A constraint is a formula ϕ defined by:

$$\phi := \quad \bot \quad | \quad t_1 = t_2 \quad | \quad R(t_1, ..., t_n) \quad | \quad \phi_1 \wedge \phi_2 \quad | \quad \neg\phi \quad | \quad \exists x \phi$$

We denote by V_ϕ the set of all free variables contain in a formula ϕ, and by D the data domain of the variables.

Terms are either variables, functions or special symbol *. Intuitively, we can see variables' value as data acquisition of sensors. Variables are defined over a data domain D, and the absence of data is represented by the symbol *. A formula ϕ, represents the constraint over the set of variable V_ϕ. We say that ϕ is satisfiable if there exists an assignment $\delta : V_\phi \to D$ such that $\delta \models \phi$. Composition of two constraints c_1 and c_2 is written as the conjunction $c_1 \wedge c_2$. If two constraints have disjoint variables, then the composition is satisfiable if and only if the two composite constraints are satisfiable. If the set of variable of two constraints intersect, satisfiability can no more be deduced from the satisfiability of composite constraints.

Example 1. We consider three constraints $east := x \neq * \wedge y = *$, $west := y \neq * \wedge x = *$ and $charge := c \neq *$, where x, y and c are three variables. We can imagine that the name of the constraints *east*, *west* and *charge* reflects to the actual behavior of our system : having the constraint *east* true results in the *east* behavior (respectively with *west* and *charge*). Since $V_{east} \cap V_{charge} = \emptyset$, if *east* is satisfiable and *charge* is satisfiable, then $east \wedge charge$ is also satisfiable. Looking at the composition of *east* and *west*, the intersection $V_{east} \cap V_{west} \neq \emptyset$ and, in this case, the composed constraint is not satisfiable.

Definition 2. *A Constraint Automaton is a tuple $\langle Q, \to, C, q_0, \rangle$ where:*

- *Q is a set of state and $q_0 \in Q$ is the initial state.*
- *C is a set of constraints.*
- *$\to \subseteq Q \times C \times Q$ is a finite relation called transition relation.*

The tuple $(q, c, q') \in \to$ represents a transition from state q to state q' subject to the constraint c. We write $q \xrightarrow{c} q'$ instead of $(q, c, q') \in \to$. If $c \equiv \bot$, we can drop $q \xrightarrow{c} q'$ from $\to$.

At this point, we should explain the behavior of a Constraint Automata $\mathcal{A}$. Before being able to do so, we must introduce a new notion. We call $\Gamma : V \rightarrow D \cup \{*\}$ the assignment map defined on the set of all free variables of $\mathcal{A}$. Given $v \in V$, $\Gamma(v)$ is either the symbol $*$ or a value from the domain D. We say that v is undefined if $\Gamma(v) = *$. Otherwise, $\Gamma(v) \in D$ is a value of the free variable v. Note that the context carried by Γ can be modified either by a local assignment (a constraint of the shape $v = d$ where $d \in D$).

It is now possible to express the behavior of the automaton in terms of sequence of constraints. Starting in the initial state q_0 with an assignment map Γ, all outgoing transition are evaluated within Γ : for all free variable v such that $\Gamma(v) \neq *$, we substitute the value $\Gamma(v)$ for v in the constraint. We then get a set of outgoing transitions whose constraints are satisfiable by Γ. We choose one constraint non deterministically, and update the assignment map Γ. We repeat this procedure in the next states.

Example 2. The automaton represents the behavior of a system oscillating between east and west non deterministically. From state q_W, three actions are possible : $west, stay_{lon}$ and $east$. If all three actions are allowed (their evaluation is true), then the choice is non deterministic. If the evaluation defines a subset $c \subset C$ of true predicate, the action is chosen non deterministically over this subset c. Constraint automata does not let us to express an order among the constraint of a subset $c \subset C$. One possible behavior of this automaton can be described by the infinite stream of constraints $\langle east, west, stay_{lon}, stay_{lon}, east, west \rangle^{\omega}$

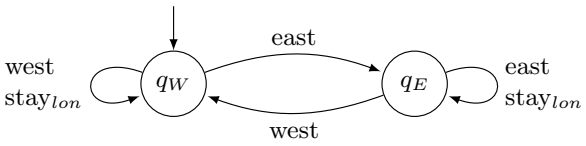

Fig. 1. Constraint automata of a moving agent

Specifying a system directly as a single Constraint Automata is not easy, since the number of states and transitions can become quite large. Instead, we want to see our system as a set of Constraint Automata in composition. We next define a composition operator for CA.

Definition 3. *Given* $A_1 = \langle Q_1, \rightarrow_1, C_1, q_{01} \rangle$ *and* $A_2 = \langle Q_2, \rightarrow_2, C_2, q_{02} \rangle$ *two constraints automata, we define the product* $A_1 \times A_2 = \langle Q, \rightarrow, C, (q_{01}, q_{02}) \rangle$ *where:*

- $Q = Q_1 \times Q_2$ *is a set of state and* $(q_{01}, q_{02}) \in Q$ *is the initial state.*
- *The transition relation* $\rightarrow$ *is the smallest relation satisfying*

$$\frac{q_{01} \xrightarrow{c_1}_1 q'_{01}, \quad q_{02} \xrightarrow{c_2}_2 q'_{02}}{(q_{01}, q_{02}) \xrightarrow{c_1 \wedge c_2} (q'_{01}, q'_{02})}$$

The assignment map Γ of $\mathcal{A}$ is the union of the assignment map Γ_1 of $\mathcal{A}_1$ and Γ_2 of $\mathcal{A}_2$. Which reflect the interaction between free variables shared between $\mathcal{A}_1$ and $\mathcal{A}_1$. Another illuminating view of the behavior of Constraint Automata in composition is given by thinking of concurrent executions. Both composite automaton tries to satisfy concurrently and consistently a constraint. In the case they succeed, they both take their transition, update the assignment map Γ and perform this new concurrent execution on the new state. It is possible to encode independent progress on each state $q \in Q$ on $\mathcal{A}$ by adding a transition (q, ϕ, q) where ϕ represents the constraint where no free variables of the automaton are defined $\bigwedge_{v \in V}(v \neq *)$.

Example 3. Example of composition

1.2 Preferences

Semirings constitutes a suitable mathematical structure to define the notion of preferences. A constraint semiring [2] induces an order relation among its element.

Definition 4. *(semiring) A semiring is a non empty set E on which operations of addition and multiplication have been defined such that:*

- $(E, +)$ *is a commutative monoid with identity element 0.*
- $(E, \times)$ *is a monoid with identity element 1.*
- *Multiplication distributes over addition from either sides.*
- $0 \times e = e \times 0 = 0$ *for all* $e \in E$

We note $\langle E, +, \times, 1, 0\rangle$ to refer to this semiring.

Definition 5. *(csemiring) A csemiring is a semiring $\langle E, +, \times, 1, 0\rangle$ with additional properties :*

- $+$ *is idempotent. We use the notation $\sum(A)$ in prefix notation to describe the sum of all elements of a possibly infinite set $A \subset E$.*
- $\times$ *is commutative.*
- $1 + e = e + 1 = 1$ *for all* $e \in E$

A csemiring admits a partial order $\leq_E$, defined as the smallest relation satisfying :

$$\frac{e, e' \in E \quad e + e' = e}{e \leq_E e'}$$

It is shown in [2] that $\leq$ satisfies the following properties:

- $\leq$ is a partial order, with minimum 0 and maximum 1;
- $x + y$ is the least upper bound of x and y;
- $x \times y$ is a lower bound of x and y;
- (S, $\leq$) is a complete lattice (i.e., the greatest lower bound exists);
- $+$ and $\times$ are monotone on $\leq$.

- if $\times$ is idempotent, then $+$ distributes over $\times$, $x \times y$ is the greatest lower bound of x and y, and (S, $\leq$) is a distributive lattice.

Intuitively, the $\times$ operator behaves as a composition operator for preferences. The $\sum$ can be seen as the choice of the best preference over a set of preferences, where 1 is the highest preference and 0 is the lowest. We give some well known instances of c-semirings.

Example 4. Examples of c-semirings :

- The Boolean semiring $\mathbb{B}$: $\langle \{\top, \bot\}, \vee, \wedge, 1_{\mathbb{B}}, 0_{\mathbb{B}} \rangle$ where $0_{\mathbb{B}} = \bot$ and $1_{\mathbb{B}} = \top$
- The Weighted semiring $\mathbb{W}$:$\langle \mathbb{N} \cup \{\infty\}, \min, +, 1_{\mathbb{W}}, 0_{\mathbb{W}} \rangle$ where $0_{\mathbb{W}} = \infty$ and $1_{\mathbb{B}} = 0$

Application In the case of our trekker, he first uses boolean semiring to model his choice. If we $left$ and $right$ are propositional variable, we could model the trekker's choice by the following value :

$$TrekChoice = left \vee right$$

Without any information, he can either chose left (make left true) or right.

Now that he has access to the distance, he can chose a weighted semiring instead, and model his choice by the following function :

$$TrekChoice = \begin{cases} left & \text{if } l +_W r = min(l, r) = l \\ right & \text{otherwise} \end{cases}$$

where l is the time it takes on the left path, and r on the right path.

2 Semiring Logic

Defining constraint predicate and c-semiring are usually orthogonal problems. In Soft Constraint Automata, boolean constraints and c-semiring are separately defined and assigned to each transition. Conceptually, we could find some justifications for this separation : the system should first look at the enable transitions (which boolean constraint is true) and chose the best transition (regarding semiring value). Enabling and ordering are two different concerns. The problem is more practical, and arises during composition. Due to the separation between constraint and c-semiring, the composition operator must differentiate both concerns. In this section, we propose a unified formal model to express both constraint and c-semiring as a soft constraint predicate. Intuitively, a soft constraint predicate is the composition of a c-semiring value from boolean semiring (e.g. the constraint), composed with another c-semiring value (e.g. the semiring value). A new composition operator is lately presented.

Definition 6. *(Language of soft constraint) A term is defined as:*

$$t := \quad v \quad | \quad f(t_1, ..., t_n) \quad | \quad *$$

A soft constraint over a csemiring $\mathbb{E}$ is a formula ϕ defined by:

$$\phi := \quad e \quad | \quad \phi_1 \times \phi_2 \quad | \quad \phi_1 + \phi_2 \quad | \quad R(t_1, ..., t_n) \quad | \quad \exists x \phi \quad | \quad t_1 = t_2$$

We denote by V_ϕ the set of all free variables contained in a formula ϕ, and by D the data domain of the variables.

Example 5. Examples of c-semirings :

Definition 7. *Csemiring Automaton (CsA) is a tuple $\langle Q, \rightarrow, \mathbb{C}, q_0, \rangle$ where :*

- *Q is a set of states and $q_0 \in Q$ is the initial state.*
- *$\mathbb{C}$ is a set of c-semiring formulas representing soft constraints.*
- *$\rightarrow \subseteq Q \times \mathbb{C} \times Q$ is a finite relation called the transition relation.*

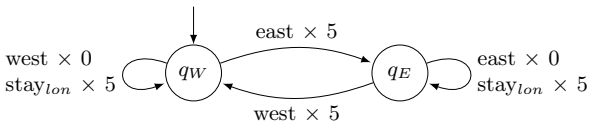

Fig. 2. Constraint automata of a moving agent

Definition 8. *Given two CsA $A_1 = \langle Q_1, \rightarrow_1, \mathbb{C}_1, q_{01} \rangle$ and $A_2 = \langle Q_2, \rightarrow_2, \mathbb{C}_2, q_{02} \rangle$, their product is $A_1 \times A_2 = \langle Q, \rightarrow, \mathbb{C}, (q_{01}, q_{02}) \rangle$ where :*

- *$Q = Q_1 \times Q_2$ and $(q_{01}, q_{02}) \in Q$ is the initial state.*
- *The transition relation $\rightarrow$ is the smallest relation satisfying*

$$\frac{q_{01} \xrightarrow{c_1}_1 q'_{01}, \quad q_{02} \xrightarrow{c_2}_2 q'_{02}}{(q_{01}, q_{02}) \xrightarrow{c_1 \times c_2} (q'_{01}, q'_{02})}$$

Example 6. The moving component can be represented as the following formula :

$$\begin{aligned}
\phi = & m = q_1 \times m' = q_0 \times west \times 5 + \\
& m = q_0 \times m' = q_1 \times east \times 5 + \\
& m = q_1 \times m' = q_1 \times east \times 0 + \\
& m = q_0 \times m' = q_0 \times west \times 0 + \\
& m = q_1 \times m' = q_1 \times stay_{lon} \times 5 + \\
& m = q_0 \times m' = q_0 \times stay_{lon} \times 5 \\
\phi = & \sum_i c_i \times w_i
\end{aligned}$$

where w_i is the weight associated to the constraint c_i.

Proposition 1. *The c-semiring automaton $\langle Q, \rightarrow, \mathbb{C}, q_0 \rangle$, where $\mathbb{C}$ is a set of boolean semiring constraint, is a constraint automaton.*

References

1. Farhad Arbab and Francesco Santini. *Preference and Similarity-Based Behavioral Discovery of Services*, pages 118–133. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
2. Stefano Bistarelli. *Semiring for Soft Constraint Solving and Programming*. Springer, 2004.
3. Tobias Kappé, Farhad Arbab, and Carolyn L. Talcott. A component-oriented framework for autonomous agents. pages 20–38, 2017.
4. Marjan Sirjani Nikunj R. Mehta and Farhad Arbab. Effective modeling of software architectural assemblies using constraint automata. 2003.