Over this document, I give an example of Rules based Automata description and composition for Component Automaton. I took the previous example of Tobias, and ignored for now semiring values. Thus, soft constraints are considered as constraint.

Rules based Automata

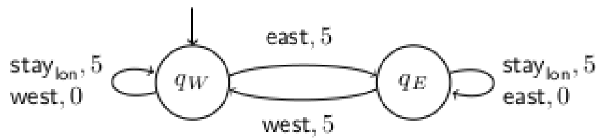

Figure 1: A component modeling a patrolling movement.

According to the Component Action System, all actions can be represented by a set of firing ports. Let's define:
west as the action triggered by the ports a, b and $\sim c$,
east as the action triggered by the ports c, b and $\sim a$,
$stay_{lon}$ as the action triggered by the port b.

Because *west* and *east* are mutually exclusive actions, "$\sim$" denotes the ports that must not fire in order for the action to be allowed. In this case, c can not fire while going *west*, and a can not fire while going *east*. Which prevent us from composing action *west* and *east*.

Now, the automaton can be viewed as a single logical formula :

$$\phi(A_1) = (a! \rightarrow west) \wedge (b! \rightarrow east \vee west \vee stay_{lon}) \wedge (c! \rightarrow east)$$

This formula says : if a fires, then *west* must be true, and if b fires, one of *east* or *west* or $stay_{lon}$ must be true, and if c fires , then *east* must be true.

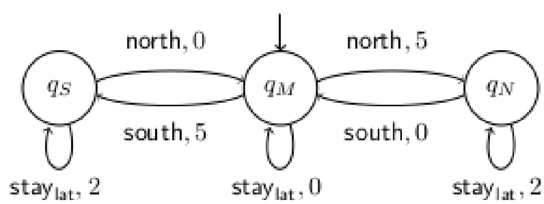

Figure 2: A component modeling divergence from the path in two directions.

The automaton A_2 of figure 2 can also be turned into a logical formula similarly as A_1.
north as the action triggered by the ports d, b and $\sim e$,
south as the action triggered by the ports e, b and $\sim d$,
$stay_{lon}$ as the action triggered by the port b.

The corresponding formula is :

$$\phi(A_2) = (d! \rightarrow north) \wedge (b! \rightarrow north \vee south \vee stay_{lon}) \wedge (e! \rightarrow south)$$

Join composition of automata from this logical perspective is a conjunction :

$$\phi(A) = \phi(A_1) \wedge \phi(A_2)$$

If we chose to develop and distribute rules :

$$\phi(A) = (a! \rightarrow west) \wedge (b! \rightarrow (east \vee west \vee stay_{lon}) \wedge (north \vee south \vee stay_{lat})) \wedge (c! \rightarrow east) \wedge (d! \rightarrow north) \wedge (e! \rightarrow south)$$

The previous formula can be written as follow :

$$\phi(A) = (a! \rightarrow r_1 \vee r_2 \vee r_3) \wedge (b! \rightarrow \bigvee_{i \in [1..9]} r_i) \wedge (c! \rightarrow r_4 \vee r_5 \vee r_6) \wedge (d! \rightarrow r_1 \wedge r_4 \wedge r_7) \wedge (e! \rightarrow r_2 \vee r_5 \vee r_8)$$

by defining
$r_1 = west \wedge north$, $r_2 = west \wedge south$, $r_3 = west \wedge stay_{lat}$, $r_4 = east \wedge north$, $r_5 = east \wedge south$, $r_6 = east \wedge stay_{lat}$, $r_7 = stay_{lon} \wedge north$, $r_8 = stay_{lon} \wedge south$, $r_9 = stay_{lon} \wedge stay_{lat}$

The main advantage of this description is the linear size (in terms of variable) of the formula after composition. The number of clauses is equal to the number of ports involved in the automaton. It's also possible to express those rules under a well chosen structure (hypergraph) such that ϕ can be represented with a linear number of rules (union of rules while composing). Several points must be considered to modelize soft component automata with this description. It's essential to integrate semiring values into the rules, i still don't fully see how to do it (turn soft constraints into constraint automata by interpreting semiring values as predicate?, ...). Besides, the composition operator needs to change to handle lexicographic composition.