1 Two operational semantics for Soft Component System

1.1 Soft Component Automata

Syntax Syntax.
State and transition, semiring values and component actions.

Example Example.
East West patrolling agent.

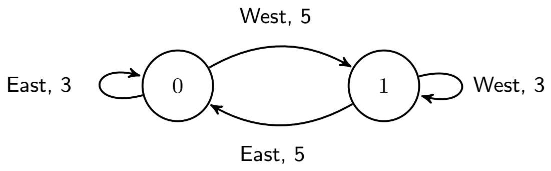

Figure 1: East-West soft component automaton

1.2 Soft Constraint Automata

Syntax Syntax.
State and transition, semiring values, synchronization constraints and data constraints.

Example Example.
East West patrolling agent.

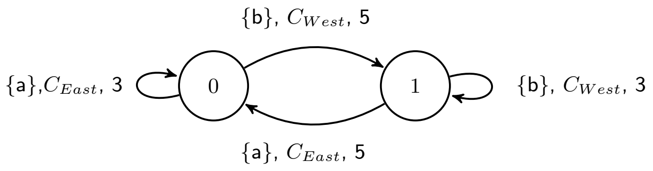

Figure 2: East-West soft constraint automaton

1.3 Soft Constraint Formula

Syntax Syntax.
Terms are ports and memory cells.

Example Example.
Given the soft constraint automaton A above, its soft constraint formula is :

$$\begin{aligned}\phi(A) = \quad & (a \neq * \wedge b = * \wedge C_{East} \wedge 3_{\mathbb{W}} \wedge s = 0) \vee (a = * \wedge b \neq * \wedge s = 0 \wedge C_{West} \wedge s' = 1 \wedge 5_{\mathbb{W}}) \vee \\ & (a = * \wedge b \neq * \wedge s = 1 \wedge C_{West} \wedge 3_{\mathbb{W}}) \vee (a \neq * \wedge b = * \wedge s = 1 \wedge C_{East} \wedge s' = 0 \wedge 5_{\mathbb{W}})\end{aligned}$$

Remark Remark.
Each clause is a transition of the one state automaton derived from the constraint automaton.

2 Compilation and Runtime

Predictability and Robustness are two main challenges involve at Soft Component System runtime.

2.1 Predictability

Semiring values make the system deterministic by defining an ordering over the compiled transitions.

Algorithm Chose transition at runtime
Semiring values define an order over a subsets of transition enabled for each system's configurations.

2.2 Robustness

Compile the transitions to a rewrite system. Prove properties on the rewrite system.

2.2.1 Compilation to Maude

Definition Terms of the rewrite system
Ports and memory cells are defined in separated functional modules.

```
op p : Nat Data* -> Fact [ctor] . *** port of the protocol
op q : Nat Data* -> Fact [ctor] . *** port of the environment
op m : Nat Data* -> Fact [ctor] .
```

Definition Rules of the rewrite system
Each clause is a rewrite rule where the left hand side consists of all the variables involved in the rule, and the right hand side consists of the all the variables involved in the rules with their updated data value.

Example Alternator

```
rl [3] : m(1,d_m1) p(3,*) => p(3,d_m1) m(1,*)  .
rl [1] : m(1,*) m(2,d_m2) => m(1,d_m2) m(2,*)  .
rl [4] : m(2,*) m(3,d_m3)  => m(2,d_m3) m(3,*)  .
rl [2] : m(1,*) m(2,*) m(3,*) p(1,d_1) p(2,d_2) p(3,*) p(5,d_5) p(7,d_7) =>
         p(1,*) p(2,*) p(3,d_1) p(5,*) p(7,*) m(1,d_2) m(2,d_5) m(3,d_7)  .

eq startc = *** initialisation of mem cells, ports data. Link between port of
                the protocol and the environment.
```

Definition Trace module
Each input port and output port add its data to the trace module. The trace represents the observable behavior of the Soft Constraint System

```
op step : String Value -> Step [ctor] .
op trace : StepList -> Fact [ctor] .
```

Definition Environment module
The environment has two rewrite rules. If the environment port is linked with an input port of the protocol, the first rule produces data. If the environment port is linked with an output port, the second rule consumes data.

```
op pg : Nat -> Data* .
op counter : Nat Nat -> Fact .

vars i j : Nat .
var n : Nat .
var k : Nat .
var d : Data .
```

```
eq pg(n) = string(n,10) .

rl [prod] : in(i) link(i,j) q(j,k,*) => in(i) link(i,j) q(j,s k,pg(k)) .
rl [cons] : out(j) link(i,j) q(i,k,d) => out(j) link(i,j) q(i,k,*) .
```

Definition Runtime module
The environment has two rewrite rules. If the environment port is linked with an input port of the protocol, the first rule produces data. If the environment port is linked with an output port, the second rule consumes data.

```
vars i j : Nat .
var k : Nat .
var d : Data .

rl [ruleOut] : link(i,j) out(i) p(i,d) q(j,k,*) => link(i,j) out(i) p(i,*) q(j,k,d).
rl [ruleIn] : link(i,j) in(i) p(i,*) q(j,k,d) => link(i,j) in(i) p(i,d) q(j,k,*) .
```

2.2.2 property analysis