Modification :
- Threshold value should be set for an automata (not for each instances of semiring).
- A function (homomorphism) has to be defined in order to keep a valid structure during semiring composition. If this function is always canonical injection, then there is no need to define it here. As soon as the homomorphism does not change the preference partial ordering, this function can be set afterwords (?).

**Soft Constraint Automata into REO semantic**

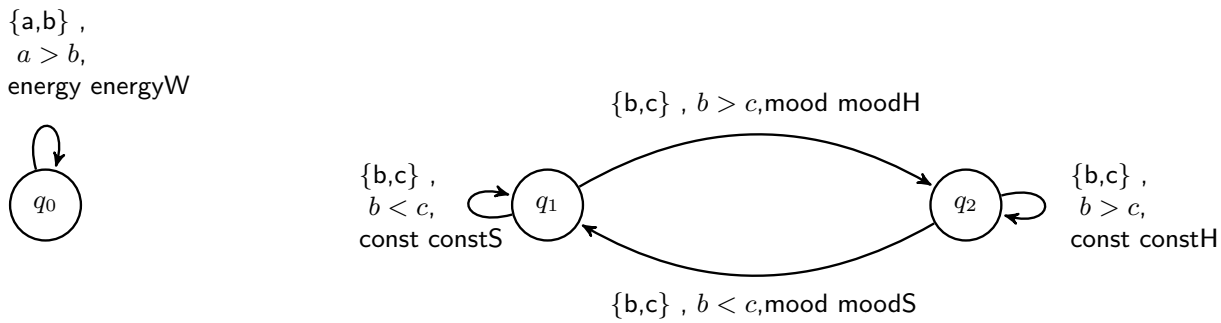


Figure 1: Automata $A_0$ on the left and $A_1$ on the right

To declare a Soft Constraint Automata within REO semantics, the name of the semirings, its parameters (such as its value and threshold) appear in the signature. Then, $A0 < energyW < x, y >> ()$ defines the automata A0 with energyW an instance of the underlying semiring energy, x the preference value given for this semiring, and y the threshold. As soon as the name of each instanciation is unique, this model lets us access to all semiring of the automata.

```
A0<energyW<2>,t=1>(){
    energy : {W}
    t : energy

    q0 -> q0 : {a,b}, a>b, energyW:{energy}

 }
```

```
A1<moodH<3>,moodS<2>,constH<0.2>,constS<0.7>,t=<0.1,1>>(){
    mood : {W}; const : {P}
    t : <const,mood>

    q1 -> q1 : {b,c}, b<c , constS:{const}
    q1 -> q2 : {b,c}, b>c , moodH:{mood}
    q2 -> q1 : {b,c}, b<c , moodS:{mood}
    q2 -> q2 : {b,c}, b>c , constH:{const}

}
```

**Composed Soft Constraint Automata into REO semantic**
Automata $A_0$ and $A_1$ can be composed to obtain a new automata $A_0.A_1$ as displayed below:

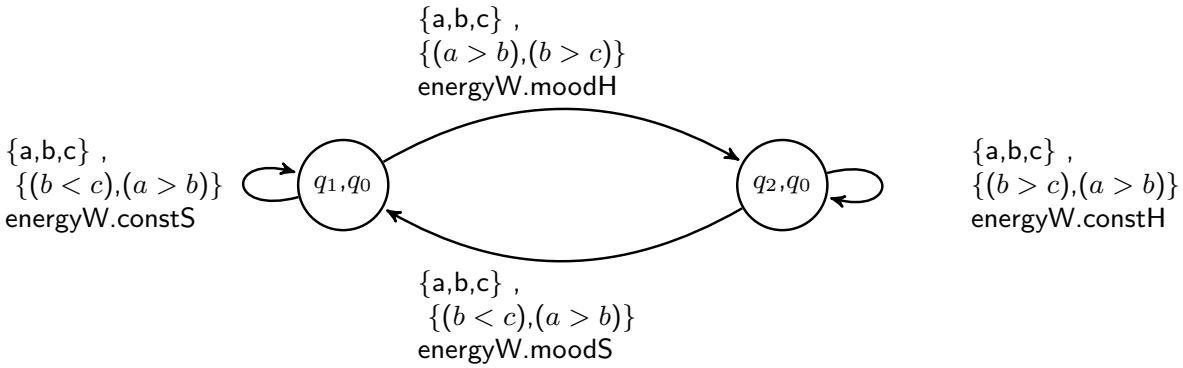


Figure 2: $A_0.A_1$ is the composition between $A_0$ and $A_1$

REO semantics for the composed automata :

```
//As it should be for composing A0 and A1 :
A2 = A0<2,t=1>.A1<3,1,0.2,0.7,t=(1,0.1)>

// Which gives :
A2 = A0.A1<energyW.constS<2,0.7>,energyW.constH<2,0.2>,energyW.moodS<2,2>,
           energyW.moodH<2,3>,t=<(1,1),(1,0.1)>>(){
    energy : {W}
    const : {P}
    t : <energy.mood,energy.const>

    (q1,q0) -> (q2,q0) : {a,b,c}, {(a>b),(b>c)}, energyW.moodH:{energy.mood}
    (q1,q0) -> (q1,q0) : {a,b,c}, {(a>b),(b<c)}, energyW.constS:{energy.const}
    (q2,q0) -> (q2,q0) : {a,b,c}, {(a>b),(b>c)}, energyW.constH:{energy.const}
    (q2,q0) -> (q1,q0) : {a,b,c}, {(a>b),(b<c)}, energyW.moodS:{energy.mood}

 }
```