Centrum Wiskunde & Informatica

A compositional formal model for cyber-physical autonomous system

Benjamin LION, Farhad ARBAB

Introduction

In this work we model a cyber-physical system as a set of interacting autonomous agents, each with its own set of actions. For instance, a drone can do several actions : take-off, turn left or right, land, discharge warning, recharge, ... These actions interact with internal or external action of the same agent, other agents, or the environment. A discharge warning will cause the drone to land; an obstacle in its path makes the drone turn right (or left). As the number of action increases, the specification of interactions among agents becomes increasingly complex. A compositional model defines a complex system as a composition of simpler components or sub-systems. Thus, a cyber physical system can be seen as a set of components, whose matching actions can compose together to form their interaction. This new system can also compose with other systems, or the environment.

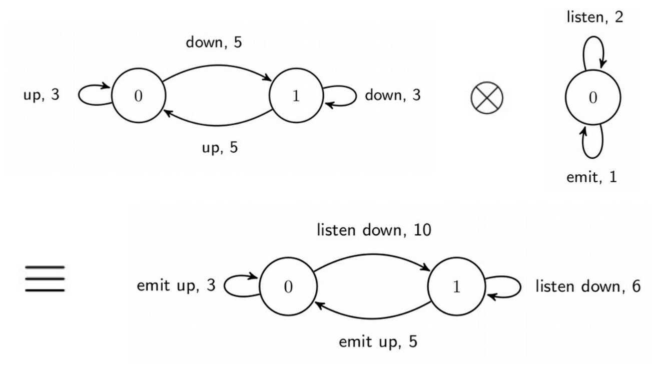
Aim

In such a compositional paradigm, agents must be capable to compromise. Each agent defines its own actions and preferences. But composing this agent with another interacting agent may yield a new set of action, and corresponding preferences. To compose preferences, we use in this work semiring valued action.

Method

A system is the composition of components, each of which may be the composition of sub-components, ...
Two semantics below represent the same example: a component with action "up" or "down" compose with a component with action "emit" or "listen". "Up" can compose with "emit" and "down" can compose with "listen". Composition lead to a compromise between both components. All action have preferences from weighted semiring $\mathbb{W}$.

Soft Constraint Automata



Remark : composition might lead to state and transition explosion.

Rule based Automata

$$(\text{down}(0,1) \wedge 5_{\mathbb{W}} \vee \text{up}(0,0) \wedge 3_{\mathbb{W}} \vee \text{down}(1,1) \wedge 3_{\mathbb{W}} \vee \text{up}(1,0) \wedge 5_{\mathbb{W}}) \quad \bigwedge \quad (\text{listen} \wedge 2_{\mathbb{W}} \vee \text{emit} \wedge 1_{\mathbb{W}})$$

$$= \quad \text{down}(0,1) \wedge \text{listen} \wedge 10_{\mathbb{W}} \vee \text{up}(0,0) \wedge \text{emit} \wedge 3_{\mathbb{W}} \vee \text{down}(1,1) \wedge \text{listen} \wedge 6_{\mathbb{W}} \vee \text{up}(1,0) \wedge \text{emit} \wedge 5_{\mathbb{W}}$$

Composition is conjunction, each clause is a transition and have a semiring values, which extend the logic : $\top = 0_{\mathbb{W}}$ and $\bot = \infty_{\mathbb{W}}$.
States, if necessary, are indicated in brackets

Code Generation

Reo is an interaction-centric concurrent programming paradigm. Constraint Automata and Rule based Automata can be expressed in Reo.
Reo to Java compiler generates executable code from a user defined semantic.
Actions are represented in external Java source files (multi threads).
Protocol models interactions and coordinates permissible actions. A structure of the code for protocol is as follow:

```
public void run() {

while (true) {

     if (Transition1.isEnabled()) {
         State.update(Transition1)
     }
…
     if (TransitionN.isEnabled()) {
         State.update(TransitionN)
     }
 }
```

Reo to Maude compiler generates rewrite rules. Each transition has its own rewrite rule.

```
rl [Transition1] : State1(variables)
                   => State2(variables) .
```

Maude opens the possibility to search through the traces of unexpected behavior.

Key points

- Simplify complex system, by using **composability** paradigm.

- Rewrite automata as **logical formulas** to counter state and transition explosion

- Extend Boolean logic with **semiring** values to express the behaviour of a system.

- Use the logic to **prove properties.**

- **Generate** executable **code**.

Conclusion

Rule based Automata is a logical model that allow compositional design of cyber-physical systems. The whole system is expressed as a single logical formula, and semiring values are now part of the logic.
One direct application is to extend Reo compiler in order to generate ordering over non deterministic actions. We also want to investigate how Rule based Automata can be used with model checkers, for error detection and diagnosis. For this purpose, a current direction is to express the composed automaton as rewriting rules in Maude, and verify properties at a meta level programming.

benjamin.lion@cwi.nl