Towards Stochastic FMI Co-simulations: Implementation of an FMU for a Stochastic Activity Networks Simulator

C. Bernardeschi[1] A. Domenici[1] M. Palmieri[2] [1]

[1]Department of Information Engineering
University of Pisa, Italy

[2]DINFO, University of Florence, Italy

2nd Workshop on Formal Co-Simulation of Cyber-Physical Systems
A satellite event of SEFM 2018, co-located with STAF 2018
June 26, 2018, Toulouse, France

A naïve question

Can I plug a new simulator into an existing multi-model?

- an almost trivial solution.

A not-so-naïve question:

Can I 'plug' a nondeterministic/stochastic simulation into a deterministic multi-model?

- yet to be found not-so-trivial solutions (see last slide).

Stochastic Activity Networks (1)

The *Stochastic Activity Networks* (SAN) are a wide-ranging and complex extension to Petri Nets.

Petri Net = places + marking + transitions + enabling conditions + firing rules.

Stochastic Petri Net = PN + stochastic transition delay.

Stochastic Activity Network = SPN + stochastic transition outcome + user-defined enabling conditions + user-defined firing rules + ...

William H. Sanders and John F. Meyer, “Stochastic Activity Networks: formal definitions and concepts”, in Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science, 2002.

Stochastic Activity Networks (2)

Activities may be *timed* or *instantaneous* (or *immediate*).

Enabling conditions: activities are enabled by user-defined **input predicates** associated with **input gates**.

An input predicate is a Boolean function of the net marking.

Firing rules: user defined functions specifying the next marking can be associated with input gates (**input functions**) and **output gates** (**output functions**).

Stochastic transition outcome: Alternative results of an activity can be specified as mutually exclusive **cases** associated with the activity.

Each case has a probability defined by a function of the marking (it may be a constant).

The Möbius tool

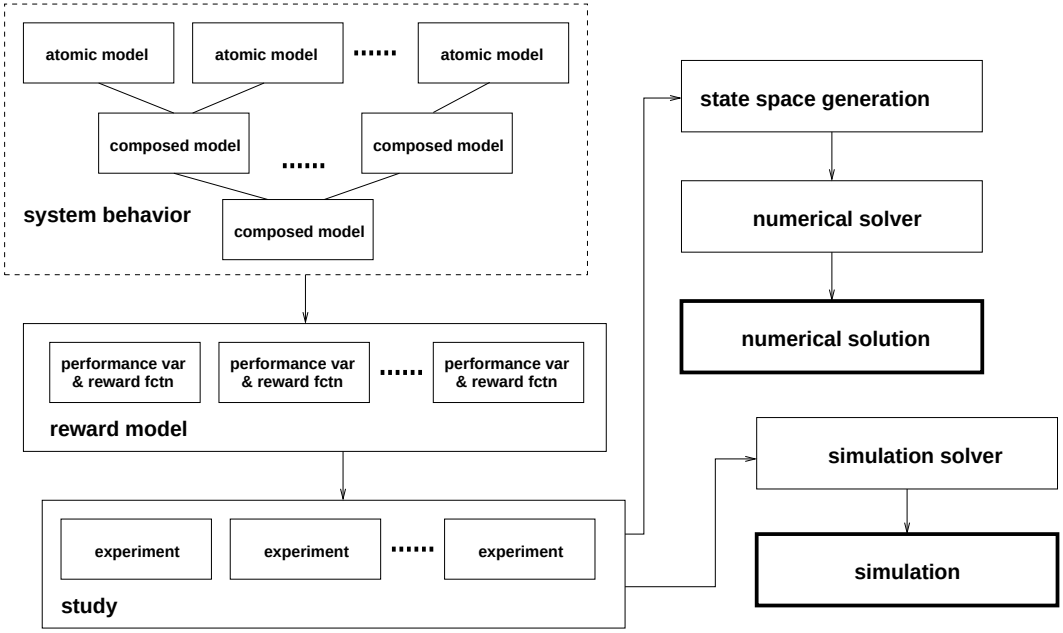
The Möbius environment provides:

- **Graphical editor** to make (*atomic*) SAN models.
- **Hierarchical composition** of models.
- **Reward models** to define and compute *performance variables*, i.e., quantitative properties related to system performance or dependability.
- **Numerical solution** of *Markov chain* equations, if certain constraints on the model are satisfied.
- **System simulation** satisfying user-defined statistical parameters, such as *confidence level* and *confidence interval*.

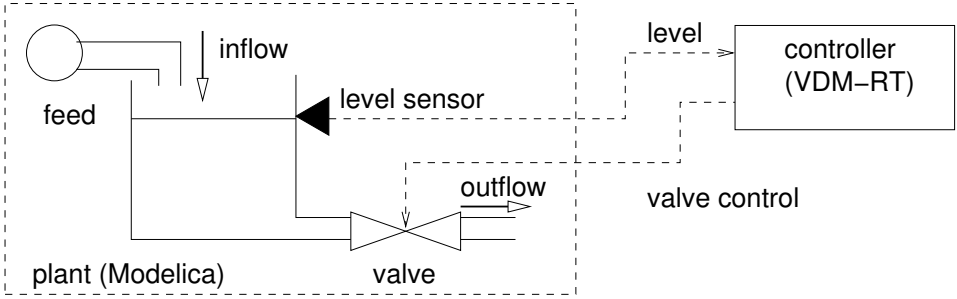
G. Clark *et al.*, “The Möbius modeling tool”, in 9th Int. Workshop on Petri Nets and Performance Models, 2001.

The Möbius Manual, Version 2.4 Rev. 1, https://www.mobius.illinois.edu/docs/MobiusManual.pdf

The Möbius analysis process



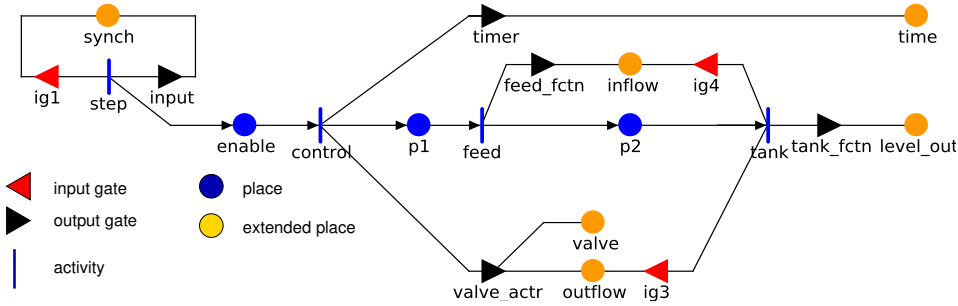
The INTO-CPS Water Tank Example



constant inflow, outflow depends on current volume

valvecontrol = if level >= maxlevel then 1.0 (**fully open**)
else if level < minlevel then 0.0 (**fully closed**)
else valvecontrol unchanged;

The SAN Model



Physical behavior:

variable feed flow (inflow)
valve opens gradually
drain flow (outflow) depends on valve area

The SAN Model: synchronization (1)

Input gate ig1 predicate:

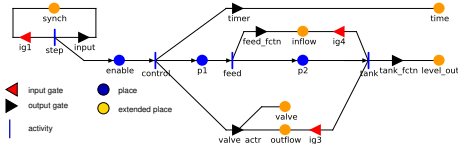
```
synch->Mark() == 1
```

Output gate input function:

```
// wait for input from controller
cin >> x;

if (x == 1) {
    valve_control.Set(1);
} else if(x == -1) {
    valve_control.Set(-1);
} else if (x == 0) {
    valve_control.Set(0);
}

synch->Mark() = 0;   // reset synch
// implicitly set enable place
```



The SAN Model: synchronization (2)

With the FMI protocol, each simulator executes one simulation step when it receives an *fmi2DoStep* request from the master algorithm. The rate at which the master algorithm issues *fmi2DoStep* requests establishes a common time base for the simulators.

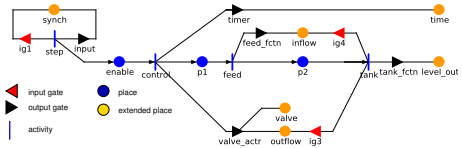
To synchronize the Möbius simulation, time has been simulated explicitly as a variable that, at each step, is incremented by a fixed amount equal to interval between *fmi2DoStep* requests.

The SAN Model: synchronization (3)

Output gate timer function:

```
time->Mark() += dt;

if (time->Mark() > maxtime)
    exit(0);
```

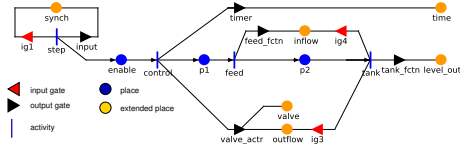


The SAN Model: valve actuation

Output gate valve_actuator function:

```
if (valve_control == 0) {
    valve->Mark() -= dt;
    if (valve->Mark() < 0)
        valve->Mark() = 0;
} else if (valve_control == 1)
    valve->Mark() += dt;
    if (valve->Mark() > 1)
        valve->Mark() = 1;
}

outflow->Mark() = C*valve->Mark();
```



The SAN Model: tank behavior

Output gate tank_fctn function:

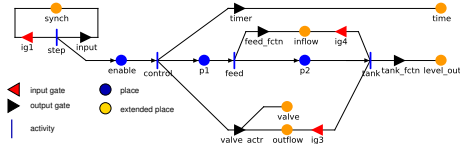
```
double netfl =
    inflow->Mark() - outflow->Mark();
level_out->Mark() =
    level_out->Mark() + netfl*dt;

if (level_out->Mark() < 0)
    level_out->Mark() = 0;

level.Set(level_out->Mark());

// set synch and enable step
synch->Mark() = 1;

// send level to controller
cout << level_out->Mark();
```



The FMU

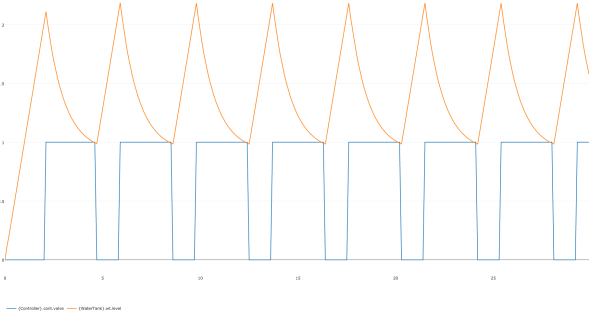
Implement fmi2Instantiate:

create pipes to redirect stdin and stdout;
spawn process running the Möbius-generated executable;
connect pipes.

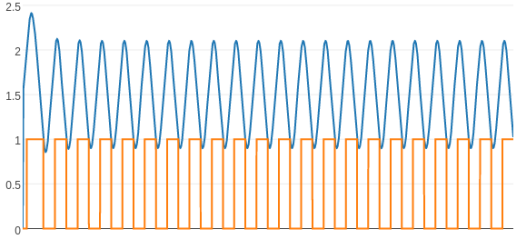
Implement fmi2DoStep:

send control signal to Möbius process;
receive and parse level value from Möbius process;
send back level value to COE.

Co-simulation



INTO-CPS distribution



Möbius tool

Further work

Initial work towards integration of statistical simulation techniques
. . . but statistics not yet used!
Much work needed to deal with important issues:

- A more modular approach is needed.
  - Reduce to a minimum the changes to fit a self-standing model into an existing multi-model;
  - how much knowledge on the model is needed (white/black/gray box)?
  - Möbius is a rather closed environment.
- A smarter synchronization mechanism is needed if statistical parameters are to be computed.
- A deeper understanding of the interaction between deterministic and non-deterministic models is needed.

Lawrence, D.P.Y., Gomes, C., Denil, J., Vangheluwe, H., Buchs, D.: Coupling Petri nets with deterministic formalisms using co-simulation. In: Proceedings of the Symposium on Theory of Modeling & Simulation. pp. 6:1–6:8. TMS-DEVS '16, Society for Computer Simulation International, San Diego, CA, USA (2016)

Thank you

Merci