Concurrent Rules Machines:

A Model of Open Cyberphysical Systems

Carolyn Talcott (joint work with Farhad Arbab
LAP September 2025

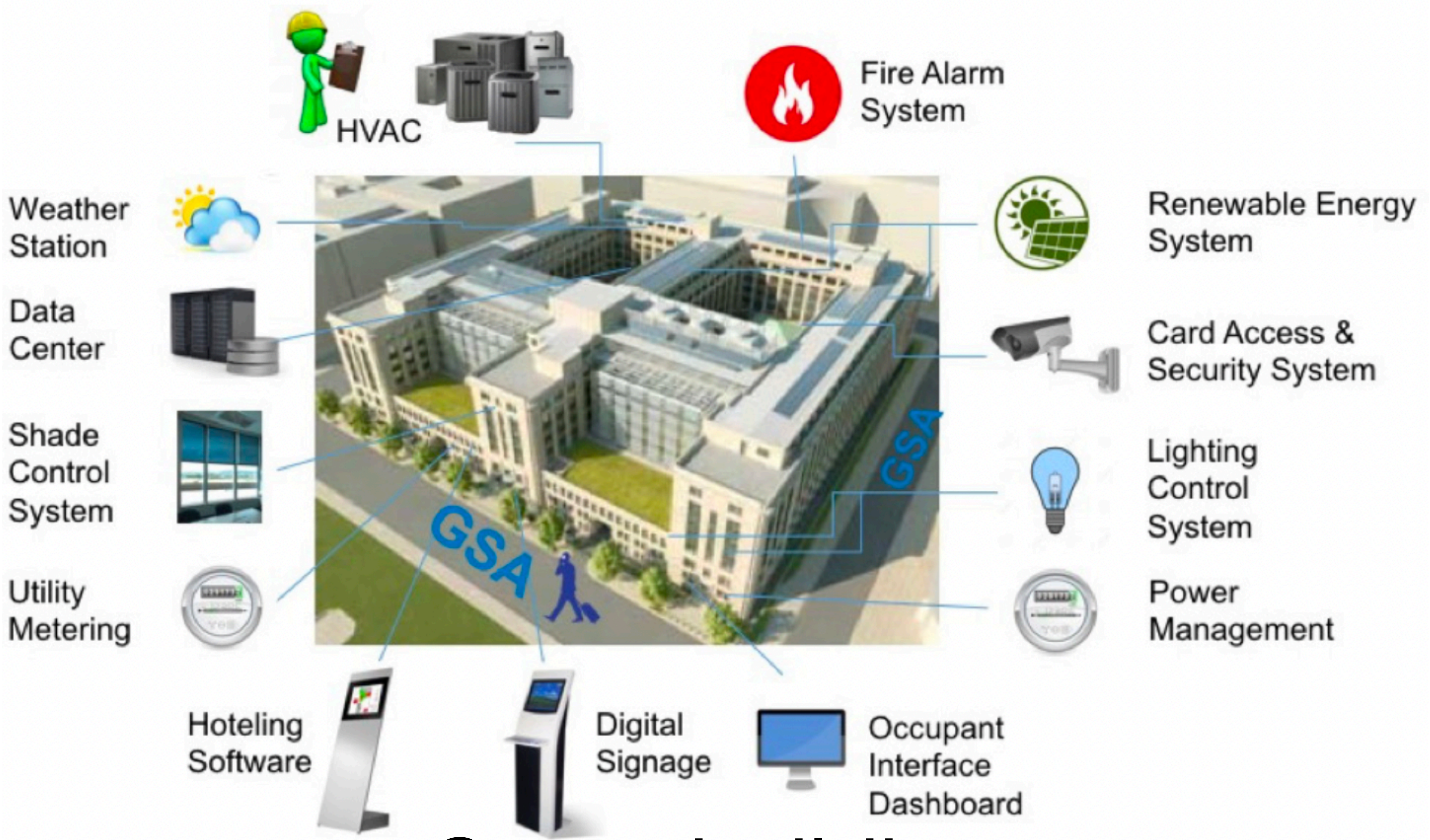
The problem

Cyberphysical systems (CPS) are increasingly present, playing critical roles.

Precision agriculture

Waymo

Smart city



Smart building

CPSs operate in unknown, unpredictable environments: effecting and being effected.

Requirements

It is important to model and reason about interaction with the environment at design time.

A foundational formal framework for developing CPS models should support representation and reasoning about:

- interaction with unknown/unpredictable environments
- discrete and continuous change (cyber and physical)
- concurrent and distributed execution
- diverse interactions among components including synchronous and asynchronous discrete interactions, as well as continuous interactions such as flow of a physical quantity among components (e.g., force, energy, liquid, etc.)
- composition operations that model these different interactions
- methods to support scaling in time and space such as symbolic reasoning and abstraction; and compositional design and reasoning

State of the Art (a sketch)

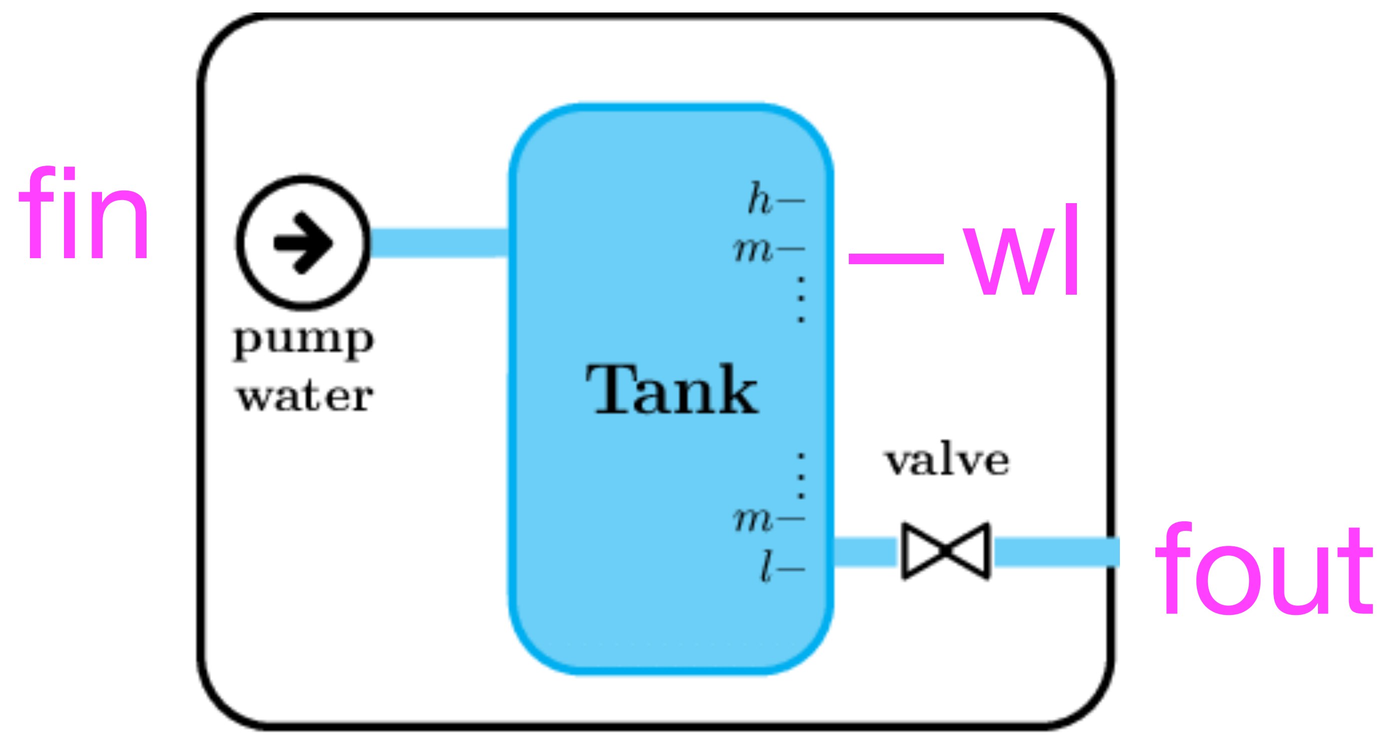
- Many kinds of automata (Team Automata, Constraint Automata, IO Automata, Interface Automata, ...) and process algebras: concurrent execution and composition with a mix of synchronous and asynchronous interaction with other components
- Rewriting logic -- a formalism for specifying concurrent and distributed systems. Rich algebra of modules for composition, diverse reasoning tools. Native execution model is closed system. Symbolic execution captures some enviornment effects
- Composable Semantics — behaviors as sets of Timed (I/O) Event Stream, composition parameterized by interaction constraints.

Contributions

- Definition of the Concurrent Rules Machine (CRM) model: structure and operational semantics.
  - Interaction with the environment is explicit, enabling representation and reasoning about open systems
  - Supports multiple models of time including implicit time (before/after relations)
- An algebra of CRM components including several composition operations, and a division relation for reasoning about decomposition.
- Symbolic execution — a step towards executability and automating reasoning

Running Example

- A Water tank with components:
  - WaterLevel (WL) -- water level sensor
  - Control (WCI/WCO) --- controls in and/or out flow
- Physical constraints:
  - $WLMin <= wl <= WLMax$
  - fin, fout in [0,1] — in/out flow rates
  - $wl(t) = wl(t0) + (fin - fout) * (t - t0)$
- Controller goal
  - $WLmin < WLMinS <= wl <= WLMaxS < WLMax$



Concurrent Rules Machines in a Nutshell

CRM Structure

Key concepts

- D -- data values
- Name -- variable names
- Valuation -- finite map, V, from Name to D
- Guard -- predicate on values of names, V |= G
- Action -- evaluates to a valuation, view as a valuation update
    - non-conflicting action sets
- Rule -- (guard, action set)

A CRM is given by a tuple (I,A_0,R,T) where

- I = (Imp, Exp, Sh) -- interface == disjoint sets of names
    - Env writes Imp -- imports
    - CRM writes Exp -- exports
    - both write/read Sh -- shared
- A_0 is a finite set of (non-conflicting) actions for initialization
- R is a finite set of Rules
- T is a finite set of (non-conflicting) actions for step termination

WL CRM

```
  WL = (WL.I, WL.A0, WL.R, WL.T)
where
  WL.I = ({'fin, 'fout, 'time}, {'wl},{})
  --- the interface with imports ('fin 'fout 'time), exports ('wl)
      and no shared variables ({}).
  WL.A0 = {(< >, \lambda (). 'wl := WCminSafe)}
     --- the initialization action
  WL.R = { (true, {a.wl})}
     --- the rule updating the water level
  WL.T = {(< 'time, 'fin, 'fout >,
          \lambda (t, fin, fout).
              {'t0 := t,'fin0 := fin, 'fout0 := fout} )}
         --- the termination action that caches the
             values of 'fin 'fout and 'time
 and
  a.wl = (< 'fin0,'fout0,'t0,'time , 'wl >,
           \lambda (fin0, fout0, t0, t, wl0).
               'wl := wl0 + (fin0 - fout0) * (t - t0) )
```

CRM Semantics

- CRM execution state is a valuation function: V : Names -f> D
- Execution alternates between action of Env and CRM rules
  - init: empty -env> V' -A_0-> V'' -T> V_0
  - step: V_j -env> V'_j -rs-> V''_j -T> V_{j+1}

where

  - V_j -env> V'_j -- Env updates some of Imp + Sh
  - V'_j -rs> V''_j -- action of rs, a set of rules st guards are satisfied by V'_j and combined actions are non-conflicting in V'_j
  - V''_j -T-> V_{j+1} -- application of actions in T
- Traces are sequences of steps

WL execution

```
time fin   fout  wl    t0  fin0  fout0
  0   1    .5                              Env initiates
                 2.0   0     1       .5    WL A0;T
  1   0    1                               Env writes
                 2.5   1     0      1      WL,R ; WL.T
  2   1    0                               Env writes
                 1.5   2     1      0      WL,R ; WL.T
  3   1    0                               Env writes
                 2.5   3      1     0      WL,R ; WL.T
  .....
```

CRM Algebra

- mtCRM = {mt,mt,mt,mt}
- C1 << C2 componentwise subset
- C1 U C2   componentwise union -- is CRM under some conditions
- C1 /\ C2 componentwise intersection -- CRM
- C1 x C2 product -- a restriction of U, guaranteed to be a CRM
- (Exists name)CRM -- hiding, makes name private
- C1 ; C2 -- a sequential composition, where at each step
  - the Env acts, then chosen rule set C1 is applied followed by T1, then chosen ruleset of C2 followed by T2
- C1 + C2 a sequential composition where at each step
  - the Env acts, then chosen rule set C1 is applied then the chosen ruleset of C2 followed by the combined actions of T1 and T2
- C1 divisibleBy C2 relation -- essentially C1 covers C2
- C1/C2 -- defined if C1 divisibleBy C2 -- essentially the set Q st C2 x Q ~ C1

CRM intersection, union, x, are associative, commutative, and idempotent.

WL division

WCO divides WCIO and WCI is in WCIO/WCO

```
      WCIO= (WCIO.I, WCIO.A0, WCIO.R, WCIO.T)
    where
      WCIO.I = ({'wl},{'fin, 'fout}, {}) --- imports, exports, shared
      WCIO.A0 = {(< >,\lambda (). 'fin :=  0; 'fout := .5)}
      WCIO.R =
       {  RI: {wc.r1, wc.r2}
          RO: { (true, {a.wo}) }
        }
      WCIO.T = {}

  WCO = (WCO.I, WCO.A0, WCO.R, WCO.T)
where
  WCO.I = ({'wl}, {'fout}, {}) -- imports,exports,shared
  WCO.A0 = {(< >,\lambda (). 'fout := .5)}
  WCO.R = { RO: (true,{a.wo}) }
  WCO.T = {}
```

What is missing?

- CRMs allow us to directly represent interaction of a system with its environment.
- One can choose different system boundaries and move components in and out using the CRM algebra.
- BUT
- How do we prove properties of specific CRMs?
- How can we automatically generate traces?
- How can we automate reasoning tasks when there is possibly unbounded non-determinism in the actions of the environment?

Idea

- Represent updates by the environment by fresh (typed) symbols (ala uninterpreted constants), possibly constrained to represent physically meaningful change or operational domain assumptions.
- CRM actions inherit the symbolic nature, with the results of actions represented by fresh symbols constrained to be equal to the action function applied to symbolic values.
- A constrained symbolic valuation represents the set of its ground instances.
- At each potential execution step we need to check that the accumulated constraints are satisfiable, i.e. the symbolic execution describes a non-empty set of actual executions.
- In this setting one can automate reachability analysis if a suitable constraint solver is available.

Some details I

- Execution state is a symbolic valuation: (*W,b*)
  - *W* maps names to symbols
  - *b* is a constraint (boolean expression) over these symbols
- The meaning of (W,b) is a set of concrete valuations is given by its instances.
  - Substitution σ is an instance of (*W,b*) if $b[\sigma]$ is true
  - V is an instance of (*W,b*) if $V = W[\sigma]$ $(= \sigma \circ W)$ for σ an instance of (W,b)

Some details II

- R is a candidate ruleset for (W,b) if
  - each rule guard is satisfied by every instance of (W,b)
  - the joint action sets of R is non-conflicting in every instance of (W,b)
- Effect of action set A on symbolic valuation
  - $A[[W]] = (Z^A, b^A)$
  - Z^A — maps write names of A to fresh symbols
  - b^A -- the conjunction of constraints for each action of A

Symbolic CRM step

$$
\begin{aligned}
(W, b) &\rightarrow_E (W^E, b \wedge b^E) \\
&\rightarrow_R (W^R, b \wedge b^E \wedge b^R) \\
&\rightarrow_T (W^T, b \wedge b^E \wedge b^R \wedge b^T)
\end{aligned}
$$

- E environment action
- R action by candidate rule set
- Termination action

Constraining the environment

- CRM steps are constrained to conform to the actions of cadidate rules and terminal actions
- The environment is constrained uniformly by a constraint specification *envB* = ($\underline{ns}$,*cs*) where *cs* is a set of assignments to and boolean expressions over names in $\underline{ns}$.
- The meaning of assignments in *envB* depend on the starting valuation and the updated valuation of a step:
    - $envB[[W^{E}(\underline{ns}),\mathrm{W}(\underline{ns})]]$
- W^{E} is *W* updated by fresh symbols for the names assigned by the environment.
- Note *envB* constrains not only the values assigned but also the actions of the environment

Example: Input controller

```
  WCI = (WCI.I, WCI.AO, WCI.R, WCI.T)

where

  WCI.I = ({'wl,'time}, {'fin}, { })  --- {Im,Ex,Sh}

  WCI.AO = {(< >, \lambda (). 'fin := finInit)}

  WCI.R = { (true, {ic.fin})}

  WCI.T = (< 'fin >, \lambda f. 'fin0 := f) --- caching 'fin

  ic.fin = (< 'fin0, 'wl >, \lambda (f0,  wl). finexp

  finexp = 'fin := (if wl > WL.smx then 0  else (if wl < WL.smn then 1 else 1/2 fi) fi) )

  envB = (<'wl 'fin 'fout 'time 't0 >,

          WL.min <= 'wl and 'wl <= WL.max and  'wl := 'wl + ('fin - 'fout) *  ('time - 't0) )

wlInit = 3   WL.max = 5  WL.min = 3  WL.smx = 9/2   WL.smn  = 7/2
```

Symbolic reasoning

- We consider two simple questions about WCI that can be answered using symbolic execution.
- Q1: Can the water level go above `WL.smx`?
- The answer is yes. (using symbolic search and SMT solver)

```
(v('fin,0) := 31/32) (v('fout,0) := 1/32) (v('wl,0) := 3)
(v('fin,1) := 1/2)   (v('fout,1) := 1/16) (v('wl,1) := 63/16)
(v('fin,2) := 1/2) (v('fout,2) := 1/8) (v('wl,2) := 35/8)
(v('fin,3) := 0)   (v('fout,3) := 1/2) (v('wl,3) := 19/4)
```

Symbolic reasoning II

- Q2: Can the water level reach `WL.mx`?
- The answer to the second question is, up to time 12, no.
- However if we change the time step to be increments of 2, then the answer is yes.

```
(v('fin,0) := 1)  (v('fout,0) := 0 (v('wl,0) := 3
(v('fin,1) := 0)  (v('fout,1) := 1/2) (v('wl,1) := 5
```

Soundness and Completeness Theorems

- **Theorem 1.** (*Soundness*) Each instance of a symbolic trace of a CRM is a (concrete) trace of that CRM

- **Theorem 2** (<u>*Restricted Completeness*</u>). If CRM, *C*, is symbolically complete, and

$$Tr = V_j \rightarrow_{F_j,R_j} V_{j+1} \mid -1 \leq j < m$$

is a concrete trace of *C* such that F j satisfies *envB* for $-1 \leq j < m$, then there is a symbolic trace, $Tr*$, of *C* with environment constraint *envB* such that *Tr* is an instance of $Tr*$.

Symbolic Completeness

- **Definition 3** (*Symbolically complete CRM*). A CRM is symbolically complete if given any (reachable) symbolic valuation (*W,b*), and substitution, σ, that satisfies *b*; if rule set *R* is a candidate for execution in the context of *W*[σ] then *R* is a candidate for execution in the context of *W*[σ′] for any σ′ satisfying *b.*

- **Lemma 1** (*One (sub)step completeness*). If CRM, *C*, is symbolically complete, (*W,b*) a reachable symbolic valuation, σ, an instance of (*W,b*), *V* = *W*[σ] the corresponding valuation instance, and *R* a candiate rule set wrt *V* then

$$V \rightarrow_R V' \;\; implies \; (W, b) \rightarrow_A (W', b')$$

where *V′* is an instance of (*W′,b′*)

Summary and Future Work

- CRM is a model of concurrent (and distributed) computation that supports
  - representing interaction with arbitrary environment
  - discrete and continuous actions
  - compositional specification via the algebra of compositions and decomposition
- Symbolic execution is step twoards automated verification
  - it is sound and complete for large class of CP CRMs
- Future work
  - Implementation in RWL
  - Constrained CRM
  - Composition via interaction constraints modeling (a)synchrony, flow, space
  - Compositional reasoning

That’s all for now !
Questions?

Some References

- Meseguer, J.: Generalized rewrite theories, coherence completion, and symbolic methods. J. Log. Algebraic Methods Program 110, 100483 (2020)

- Arbab, F., Talcott, C.: Concurrent rules machines. In: Rebeca for actor analysis in action: essays in the honour of Marjan Sirjani LNCS Festschrift, vol. 15560. Springer (2025).

- Arbab, F., Talcott, C.: Open CPS: a Symbolic Model. In: Concurrent Programming, Open Systems and Formal Methods LNCS, vol. 16120. Springer (2025). ¨