An integrated approach to P systems formal verification

Marian Gheorghe[1,2], Florentin Ipate[2],

Raluca Lefticaru[2], Ciprian Dragomir[1]

[1]University of Sheffield

[2]University of Pitesti

Summary

•“Integrated” formal verification approach

• Steps in formally verifying basic P systems

•Transforming a P systems into a NuSMV specification (through a Kripke structure)

•Extracting properties from P-lingua traces

•Verifying properties

Steps in formally verifying a P system

Given a one-membrane P system, *Π*, build up the following steps

• Kripke structure – M_{Π} associated with *Π;* translating the rules and the semantics of the *Π* to M_{Π}

• specify – M_{Π} in NuSMV; states, transitions and transformations are generated

•extract properties – from P-lingua simulations extract invariants; first, using P-lingua simulations, traces of execution are obtained and then properties extracted using Daikon

•query – the NuSMV system by using LTL statements; properties regarding the system are formulated

Kripke structure

$$M = (S, H, I, L)$$

where S – finite set of states; $I \subseteq S$ – initial states; $H \subseteq S \times S$ is a left-total transition relation (left-total - $\forall s \in S, \exists s' \in S$, such that $(s,s') \in H$); L is an interpretation functions associating to each state a set of atomic propositions true in that state.

In general a system with variables $var_1, \ldots, var_k$, and Val_i the set of values for var_i has the set $S=\{(v_1, \ldots v_k) \mid v_i \in Val_i\}$, and $AP=\{(var_i=v_i) \mid v_i \in Val_i , 1 \leq i \leq k\}$.

In what follows three types of states are built: normal, final and halt (sink) states.

Kripke structure associated with a P system

Given $\Pi=(V, \mu, w, R)$ - one-membrane P system with V having k symbols and R containing simple rewriting rules r_i: $u_i \rightarrow v_i, 1 \leq i \leq m;$ the multisets will be recorded as vectors of integers $u \in N^k$.

The Kripke structure M_Π associated with Π utilises two predicates
$MaxPar(u, u_1, v_1, n_1, \ldots u_m, v_m, n_m), u \in N^k, n_i \in N, 1 \leq i \leq m$ and
$Apply(u, v, u_1, v_1, n_1, \ldots u_m, v_m, n_m), u,v \in N^k, n_i \in N, 1 \leq i \leq m.$

MaxPar means a computation from u develops in maximally parallel mode, r_i: $u_i \rightarrow v_i$, applied $n_i \geq 0$ times, $1 \leq i \leq m$ to u.
Apply means that v is obtained from u.

– Dang, Ibarra et all, 2006

NuSMV specification – maximal parallelism

Let $\Pi=(V, \mu, w, R)$, where, $V=\{a,b,c,d,x,y\}$, $w=xy$, R contains r_1: $x \rightarrow a$, r_2: $y \rightarrow b$, r_3: $a \rightarrow xc$, r_4: $b \rightarrow ydd$

MaxPar predicate = for each rule the number of symbols occurring on the left hand side are consumed in a maximal way (if t designs the total number of symbols available and $next(n_i)$ the number of times r_i is applied in a maximal way, then $t\text{-}next(n_i)=0$). So, for the above P systems the conditions

$x\text{-}next(n_1)=0$ & $y\text{-}next(n_2)=0$ & $a\text{-}next(n_3)=0$ & $b\text{-}next(n_4)=0$

Additional conditions characterise states and transitions.

NuSMV specification – states & transitions

Let $\Pi=(V, \mu, w, R)$, where, $V=\{a,b,c,d,x,y\}$, $w=xy$, R contains $r_1: x \to a$, $r_2: y \to b$, $r_3: a \to xc$, $r_4: b \to ydd$

Apply predicate = requires to identify states and transitions (to get a finite number of states, the multisets are restricted to a finite set).

In a previous observation we mentioned three types of states – normal, final and halt.

All normal states will be compacted in one state called running (i.e., it contains all the values of the multisets u, that are within the limits chosen, $|u| \leq Max$, no of rewritings in a step $\leq MStep$).

NuSMV specification – states & transitions (2)

Let $\Pi=(V, \mu, w, R)$, where, $V=\{a,b,c,d,x,y\}$, $w=xy$, R contains
$r_1: x \rightarrow a$, $r_2: y \rightarrow b$, $r_3: a \rightarrow xc$, $r_4: b \rightarrow ydd$

state = running & next(state) = running & -- next state

$next(x) = x - next(n_1) + next(n_3)$ & -- next multisets, x

$next(y) = y - next(n_2) + next(n_4)$ & -- y

$next(a) = a - next(n_3) + next(n_1)$ & -- a

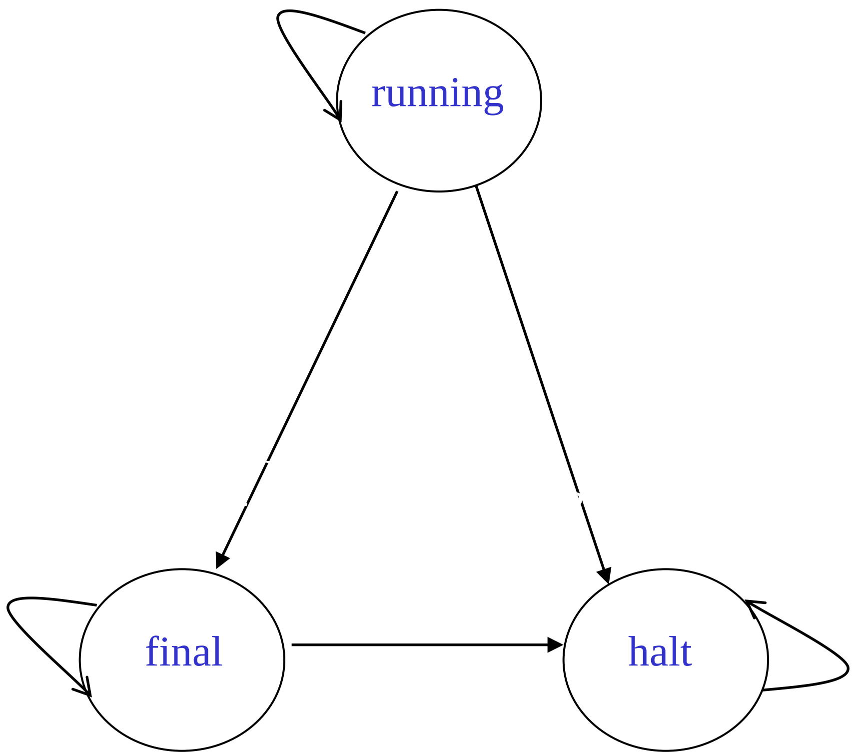
$next(b) = b - next(n_4) + next(n_2)$ & -- b

$next(c) = c + next(n_3)$ & -- c

$next(d) = d + 2*next(n_3)$ -- d

... -- conditions to stay within running

M_{Π} diagram



running – $\{u|\ |u| \leq Max$, and no more than *MStep* writings$\}$;

halt – abnormal behaviour: an u, is obtained such that $|u| > Max$ or $> MStep$ writings used

final – terminal step occurs; *MaxPar* has all $n_i = 0$

P-lingua traces and invariants extraction

- For a (basic) P system represented in P-lingua execution traces are obtained – values of the multisets
- Conversion to Daikon inputs
- Extraction of invariants and other properties (pre- and post-conditions)
- Tools utilised

Example 1

Let $\Pi=(V, [], w, R)$, where, $V=\{a,b,c,d,x,y\}$, $w=xy$, R contains $r_1: x \to a$, $r_2: y \to b$, $r_3: a \to xc$, $r_4: b \to ydd$

A computation

$xy \Rightarrow ab \Rightarrow xcydd \Rightarrow acbdd \Rightarrow xccydddd \Rightarrow \ldots\ xc^n yd^{2n} \Rightarrow ac^n bd^{2n}\ \ldots$

Invariants identified

$2*c - d == 0$ $(2*orig(c) - orig(d) == 0)$

a is one of $\{0, 1\}$ – similar for b, x, y

$c == 0 ==> orig(c)==0$ – consequence pattern; similarly for d

In NuSMV these can be verified by *G((c=0)->(c_old=0))* etc.

Other types of P systems

- A (basic) P system working in asynchronous mode (if Π works asynchronously then

$$next(n_1) + next(n_2) + next(n_3) + next(n_4) > 0)$$

i.e., at least one rule is applied; the transitions remain the same.

- When electrical charges are used then the maximal parallelism is restricted to the rules available for specific charge values.

- When more than a compartment is utilised then a suitable codification for objects is applied.

Example 2

Let $\Pi_1=(V, [[]_2]_1, xy, \lambda, R)$, where, $V=\{a,b,c,d,x,y\}$, R contains
$r_1: x[]^0_2 \rightarrow [a]^+_2$, $r_2: y\ []^0_2 \rightarrow [b]^+_2$, $r_3: [a \rightarrow xc]^+_2$,
$r_4: [b \rightarrow ydd]^+_2$, $r_5: [x]^+_2 \rightarrow x[]^0_2$, $r_6: [y]^+_2 \rightarrow y[]^0_2$

A computation in Π_1 is very similar to the one in Π, but it uses two compartments and electrical charges.

If we run either Π or Π_1 in an asynchronous way then

*2*c – d == 0 (2*orig(c) - orig (d) == 0)*

is no longer true, whereas

a is one of *{0, 1}* – similar for *b, x, y*

c == 0 ==> orig(c)==0 – consequence pattern; similar for *d*

remain valid and verifiable by NuSMV.

Example – predator-prey

The non-deterministic variant, $\Pi_{PP}=(V, [], w, R)$, where, $V=\{a,b,x,y\}$, $w=a^{100}x^{100}y^{10}$, R contains r_1: $ax \rightarrow xx$, r_2: $xy \rightarrow yy$, r_3: $y \rightarrow b$

Invariants identified and proven by NuSMV

b == 0 ==> orig(b)==0

orig(a)==0 ==> a== 0

Obs. In the non-deterministic case there are no general oscillatory processes that can be revealed.

Achievements and drawbacks

- Previous approach on model checking stochastic P systems has been now extended to generic classes of P systems with maximal parallelism.
- Basic properties are found using Daikon and proved by NuSMV.
- Both are integrated within some tools that include P-lingua as well.
- Daikon fails to reveal more complex functions.
- NuSMV does not scale up well.
- Other model checkers can be utilised (work on SPIN is under consideration).

Questions?