16th International Conference on Reachability Problems

OCTOBER 17-19 2022, KAISERSLAUTERN, GERMANY

Canonization of Reconfigurable PT Nets in Maude

Lorenzo Capra

Dipartimento di Informatica, Università degli Studi di Milano - Italy

Motivations

Dynamic reconfiguration is key feature of modern distributed systems:

- automated, self-adaptive, reactive, FT...

Need for well-supported, easy-to-use formal models.

- weakly supported by tools

Motivations

Dynamic reconfiguration is key feature of modern distributed systems:

- automated, self-adaptive, reactive, FT...

Need for well-supported, easy-to-use formal models.

- weakly supported by tools
- sound but restrictive style (e,g., pushout im GTS)

DIPARTIMENTO DI INFORMATICA

Motivations

Dynamic reconfiguration is key feature of modern distributed systems:

- automated, self-adaptive, reactive, FT...

Need for well-supported, easy-to-use formal models.

- weakly supported by tools
- sound but restrictive style (e,g., pushout im GTS)
- expressive not enough to specify structural changes (PN, PA)

DIPARTIMENTO DI INFORMATICA

Motivations

Dynamic reconfiguration is key feature of modern distributed systems:

- automated, self-adaptive, reactive, FT...

Need for well-supported, easy-to-use formal models.

- weakly supported by tools
- sound but restrictive style (e,g., pushout im GTS)
- expressive not enough to specify structural changes (PN, PA)
- richly annotated (user-unfriendly, e.g., PN extensions)

Proposal

Efficient `Maude` implementation of "rewritable" PNs

- `Maude`: expressive, efficient, purely declarative, reflective, pattern matching modulo-associativity, Rewriting Logic semantics. Logical framework for other formalisms.
- PN: reference model for concurrent/distributed systems

Proposal

Efficient `Maude` implementation of "rewritable" PNs

- `Maude`: expressive, efficient, purely declarative, reflective, pattern matching modulo-associativity, Rewriting Logic semantics. Logical framework for other formalisms.
- PN: reference model for concurrent/distributed systems
  - distributed state notion (marking)

Proposal

Efficient `Maude` implementation of "rewritable" PNs

- `Maude`: expressive, efficient, purely declarative, reflective, pattern matching modulo-associativity, Rewriting Logic semantics. Logical framework for other formalisms.
- PN: reference model for concurrent/distributed systems
  - distributed state notion (marking)
  - wide range of validation/analysis techniques

DIPARTIMENTO DI INFORMATICA

PT nets & Maude

PT system (with inhibitor edges) $S := (P, T, I, O, H, m_0)$, where $P \cap T = \emptyset$, $\{I, O, H\} : T \to Bag[P]$, $m_0 \in Bag[P]$.

$t \in T$ is *enabled* in m iff: $I(t) \leq m \wedge H(t) >' m$

It may *fire*, leading to $m' = m + O(t) - I(t)$

Interleaving semantics of S: *reachability graph* (Nodes: reachable markings. Edges: direct state-transitions).

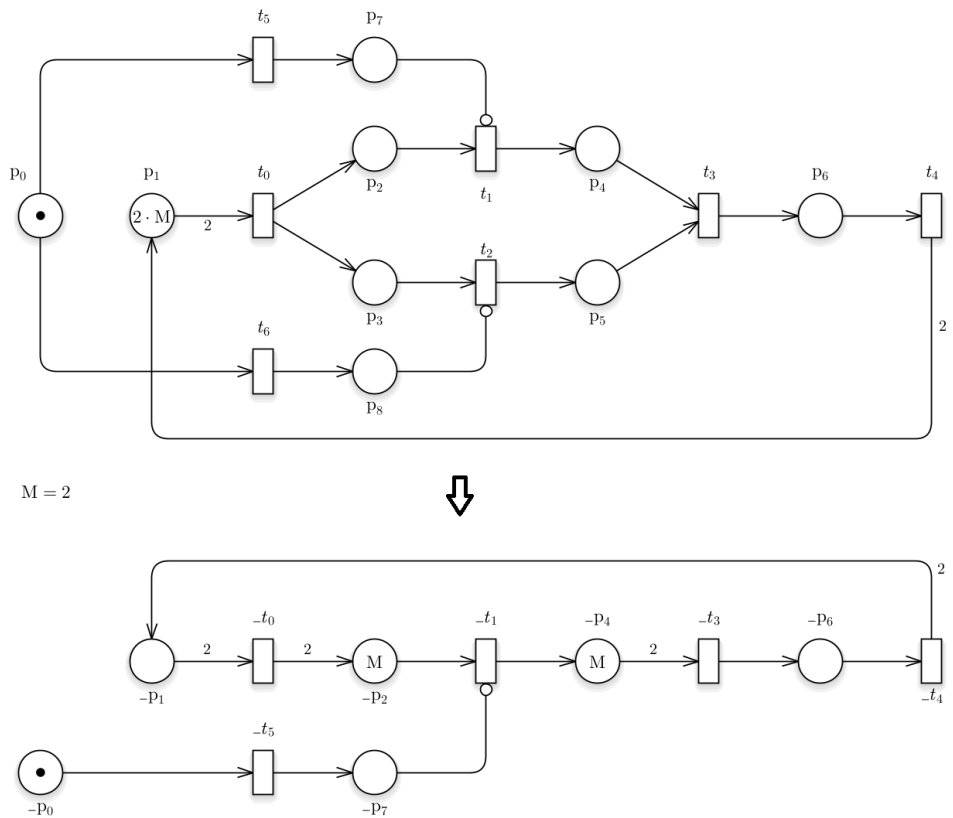
Intuitive rewriting semantics.

- Maude program: Functional modules (only equations) + System modules (rules [+ equations]).
- *Functional* module: *equational theory* $(\Sigma, E \cup A)$ in membership equational logic. Model: *Initial algebra* ($T_{\Sigma/E\cup A} \cong CAN_{\Sigma/E\cup A}$).
- *System module*: *rewrite theory* $\mathcal{R} = (\Sigma, E \cup A, \phi, R)$. R: set of *rewrite rules* (ϕ: operator frozen arguments).

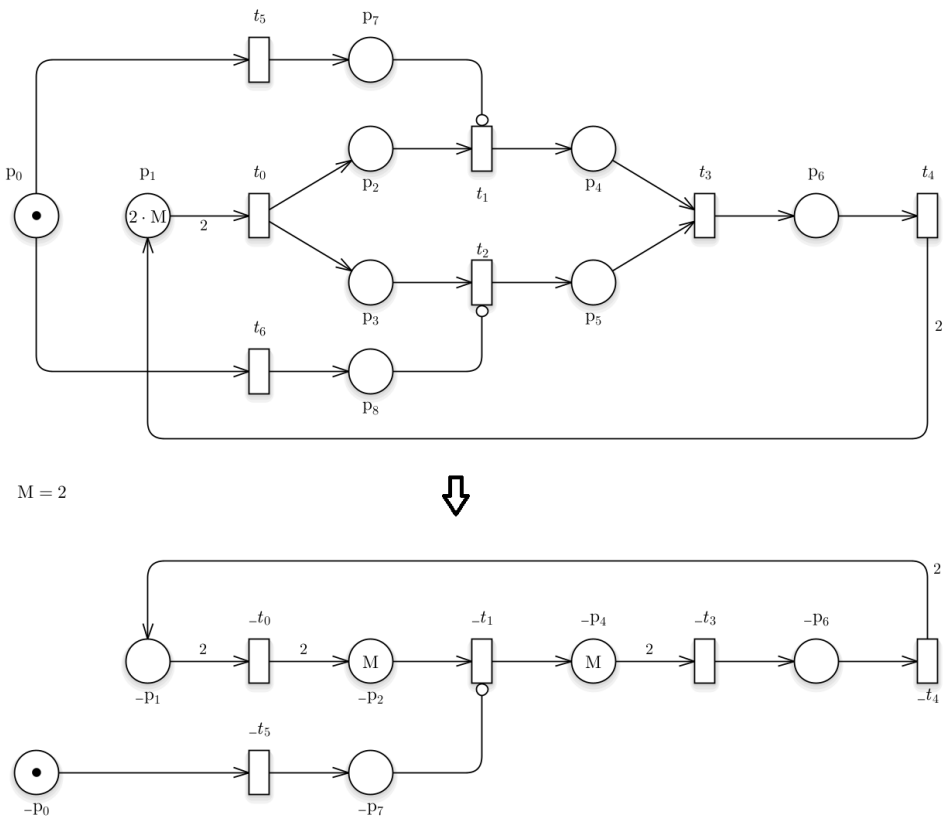
  Model of $\mathcal{R}$: labeled transition system. For each kind k.

  States: $T_{\Sigma/E\cup A,k}$. Transitions : $[t] \stackrel{[\alpha]}{\rightarrow} [t']$, with $[t], [t'] \in T_{\Sigma/E\cup A,k}$ ($[\alpha]$: equivalence class of rewrites).

Running Example: Gracefully Degrading PL

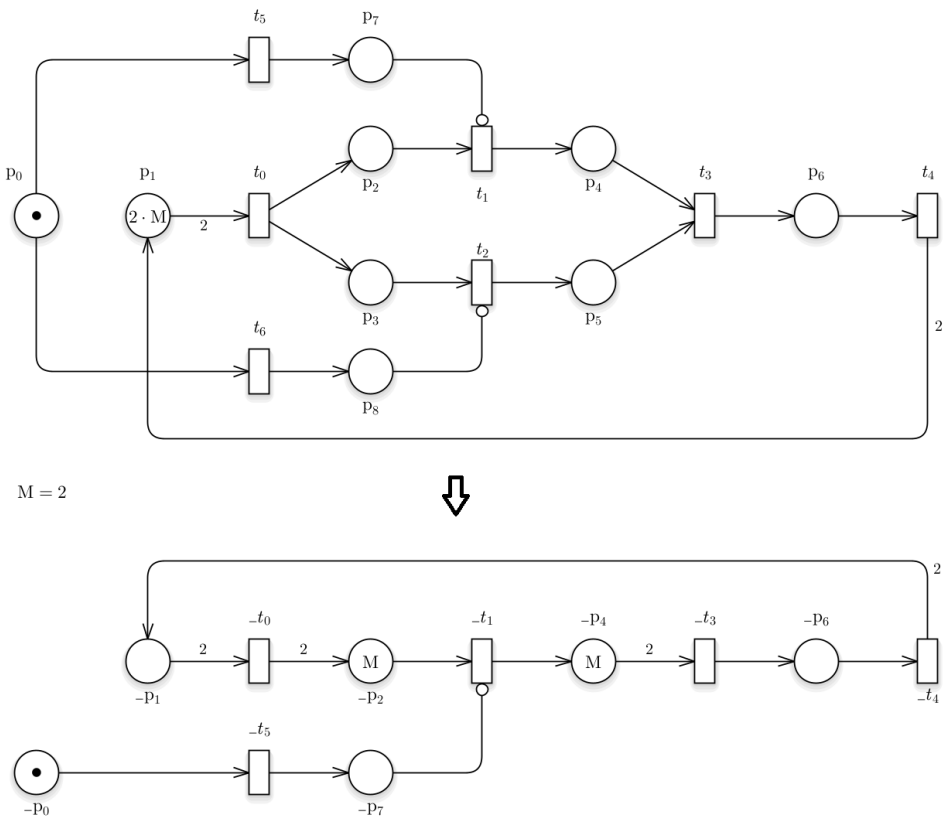


Running Example: Gracefully Degrading PL



- Raw pieces are worked by two symmetric lines.
- Upon a fault, the PL layout changes to continue working with the left line

Running Example: Gracefully Degrading PL



- Raw pieces are worked by two symmetric lines.
- Upon a fault, the PL layout changes to continue working with the left line
- Challenging: safely moving raw pieces from one line to the other

Maude specification of Rewritable PT nets

Builds on: `BAG{X :: TRIV}`, `MAP{X :: TRIV,Y :: TRIV}`

`2 . p(1) + 1 . p(2) + 1 . p(3)`

Place-bag (associative weighted sum).

`t(0) -> [2 . p(1), 1 . p(2) + 1 . p(3), nilP] , t(1) |-> [1 . p(2) , 1 . p(4), 1 . p(7)]`

Net (associative map from transitions to triplets of P-bags)

`t(0) -> [2 . p(1), 1 . p(2) nilP] , t(1) |-> [1 . p(2) , 1 . p(4), 3 . p(7)] 3 . p(1)`

System (juxtaposition of a Net and a P-bag).

Dynamics of a Maude Rewritable PT System

```
mod PT-EMU is *** PT dynamics
 pr PT-SYS .
 var T : Tran .
 var I O H S : BagP .
 var N N' : Net .
 crl [firing] : N S => N S + O - I if T |-> [I,O,H], N' := N /\ I <= S /\ H >' S . *** PT firing rule
endm

mod RWPT-FMS is *** specification of PL reconfiguration (two rules)
 pr PT-FMS .
 vars N N' : Net .
 vars Tload Tfail1 Tfail2 Tass Tline1 Tline2 Trest : Tran .
 vars P0 P1 P2 P3 P4 P5 P6 P7 P8 PF : Place .
 var S : Pbag .
 var K : NzNat .
 crl [r1] : N S + 1 . P8 => ( N' ; Tload |-> [2 . P1, 2 . P2, nilP] ; Tass |-> [2 . P4, 1 . P6, nilP])
  set(S, P3, 0) + S[P3] . P2 + 1 . P0 if (N' ; Tload |-> [2 . P1, 1 . P2 + 1 . P3, nilP ] ; Tline2 |-> [1 . P3, 1 . P5, 1 . P8] ;
  Tass |-> [1 . P4 + 1 . P5, 1 . P6, nilP ] ; Tfail2 |-> [1 . P0, 1 . P8, nilP]) := N /\ dead(N S + 1 . P8) .
 crl [r2] : N S + 1 . P7 => N' (set(set(S, P1, S[P1] + S[P2] + S[P4]), P2, 0)) - 1 . P4
  if (N' ; Tfail1 |-> [1 . P0, 1 . P7, nilP] ; Tload |-> [2 . P1, 2 . P2, nilP] ; Tline1 |-> [1 . P2, 1 . P4, 1 . P7] ;
  Tass |-> [2 . P4, 1 . P6, nilP] ; Trest |-> [1 . P6 , 2 . P1, nilP ]) := N /\ N' =/= emptyN /\
  dead((Tload |-> [2 . P1, 2 . P2, nilP] ; Tass |-> [2 . P4, 1 . P6, nilP] ; Trest |-> [1 . P6 , 2 . P1, nilP ]) S) .

endm
```

Property verification

1. Maude's model-checking facilities. E.g.: Final states in the whole system including reconfiguration:

```
search net m0 =>! X:System .
search in RWPT-FMS : net m0 =>! X:System .
Solution 1 (state 460)
states: 486 rewrites: 34576 in 40ms cpu (40ms real) (864400 rewrites/sec
X:System --> ...
...
```

2. Duality: Structural analysis based on PT incidence matrix: semiflows (invariants), structural relations, etc.

Issues

Does the approach scale up?

- As usual in graph rewriting, we should reason up to isomorphism.
- Two kinds of *symmetries*: those due to a PT system's dynamics (equivalent markings) and those caused by its evolution
- Classical techniques based on symmetry detection in (High-Level) PN do not work
- Possible approach: *canonization* of `System` terms seen as *labelled graphs*, so that[1]

$$S \cong S' \;\Leftrightarrow\; canonize(S) == canonize(S')$$

$$rl\ s \Rightarrow canonize(s')$$

[1] GI s thought neither P nor NP-complete. Maybe quasi-polynomial

Canonization Algorithm (purely Maude)

- List-views of bags are ordered to get a unique minimal representative for a System term. Basis: `op swap : System Place Place -> System .`
- Rules have to met a *symmetric form* (syntactically characterized) for the reduced reachability graph to be a quotient of the ordinary RG for which strong-bisimulation holds.

```
fmod CAN-PT-SYS is
 protecting PT-SYS .
 op canonize : System -> System . *** top operator
 op regroup : System -> ListOfList{Limatrix} . *** partitions adjacency matrices
 op $reorder : NeSetOfListOfList{Limatrix} -> NeSetOfListOfList{Limatrix} .
 op canonizeP : NeListOfList{Limatrix} Place NePlist ->
                    NeSetOfListOfList{Limatrix} .
 op $candidatesToswap : ListOfList{Limatrix} Place -> Pset .
 ...
endfm
```

Why implementing canonization in Maude

- Well-known graph-canonization algorithms (e.g., Nauty/Traces, Bliss, etc.) work or unlabelled or partially labelled graphs (further encoding/decoding needed).
- Linking Maude's interpreter to external tools is technically complex .
- Exploitation of the efficient algebraic representation of PT nets .
- Full control: heuristics (pruning computation branches), structural optimizations (see ongoing work) .

Experimental evidences

Table 1: Performance of `search` as M (pieces) varies (single PL) – time in sec

M	(ord.) states	time	(can.) states	time
2	116	0.004	64	0.200
4	535	0.043	286	1.010
8	3615	0.330	1891	7.539
16	33303	3.917	17137	76
32	383911	57.710	195229	1312

Experimental evidences (follows)

Table 2: Performance of `search` as N (PL replicas) varies

N	(ord.) states	time	(*can.*) states	time
1	116	0.004	64	0.200
2	1532	0.15	402	1.36
3	14147	2.48	1139	7.05
4	114611	34	2485	68
5	869090	466	5622	557
6	6307010	7453	10351	6185

Open issues/ongoing work

- ☐ The canonization works well in case of irregular PT structures. Less efficient when symmetric components are present.
- ☐ Using a more structured labelling (outlining automorphic-equivalent nodes) and symmetry-preserving net operators looks very promising.

  op `par` : `System NzNat String` −> [`System`] . *** n isomorphic replica

  Slight modifications to the current version of the algorithm. The time for $N = 6$ decreases of more than a magnitude order.
- ☐ Mixed use of (equation) *abstractions* and canonization (future work).

Thank you for the attention!

https://github.com/lgcapra/rewpt