Translating Statecharts to behaviourally equivalent Timed Petri Nets

Matteo Guastella
matteo.guastella@student.uantwerpen.be

University of Antwerp

Overview

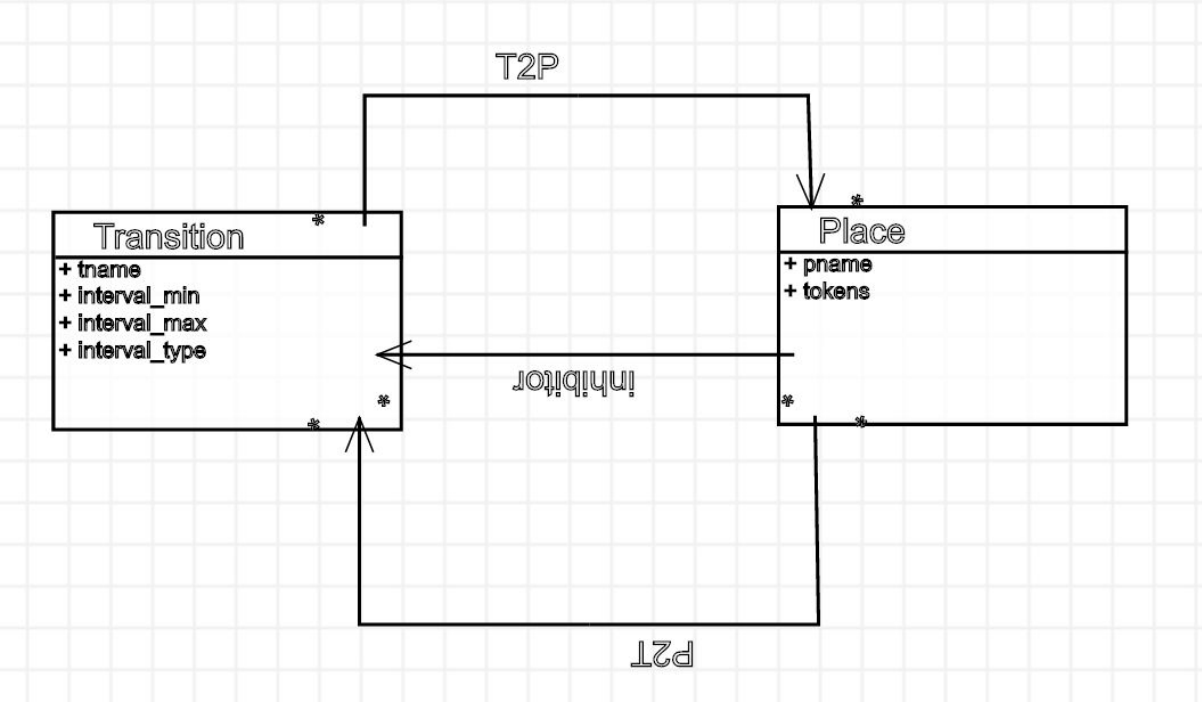
- Goals
- Timed Petri Net formalism in AToMPM
- Boundaries
- Transformation by examples
- Put all together
- Exporting to TINA toolbox
- Future work

Goals

- From StateCharts to Timed Petri Nets:
  - Create timed petri net formalism in AToMPM
  - Rule based transformation between StateCharts and TPN
  - Exporting TPNs to TINA toolbox
  - Assess the goodness of the transformation

Timed Petri Net in AToMPM

Abstract Syntax

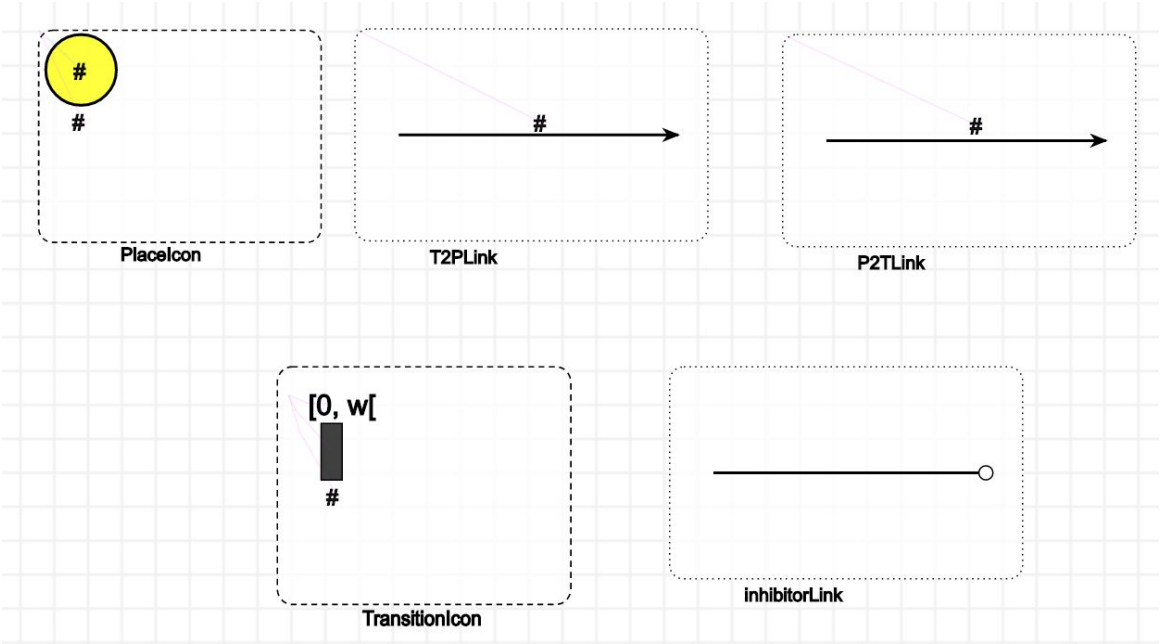


Note:

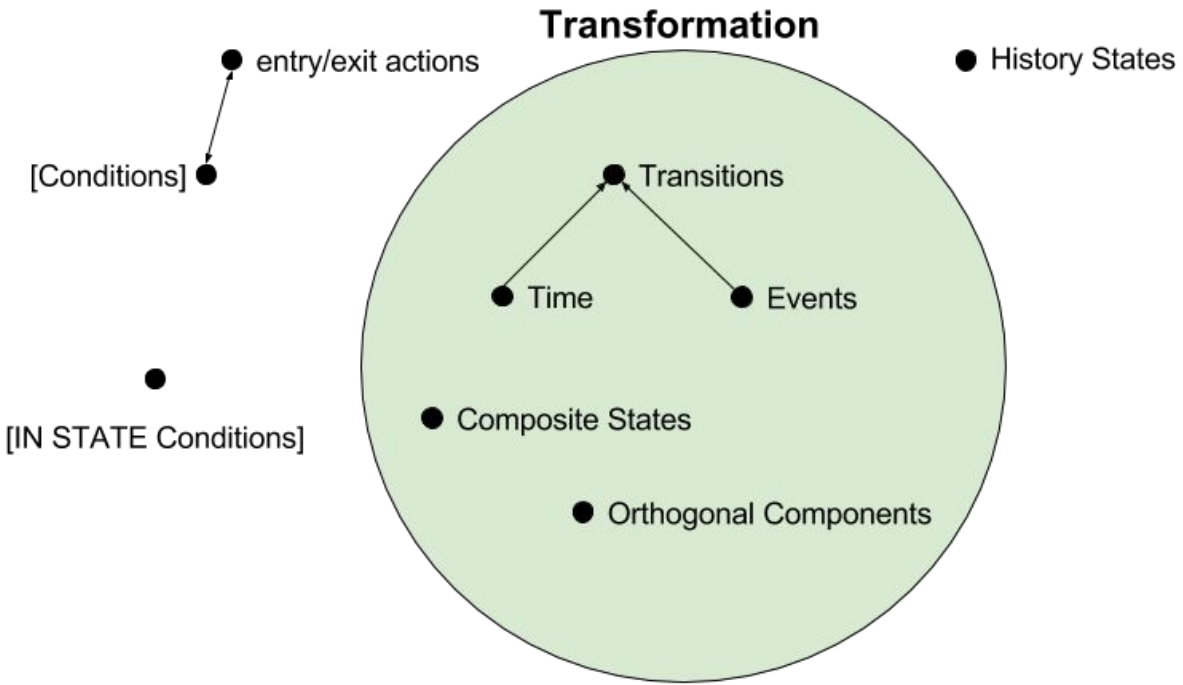
- -1 = inf. = w
- interval_type:
  - [ ]
  -] [
  - [[
  -]]

Timed Petri Net in AToMPM

Concrete Visual Syntax

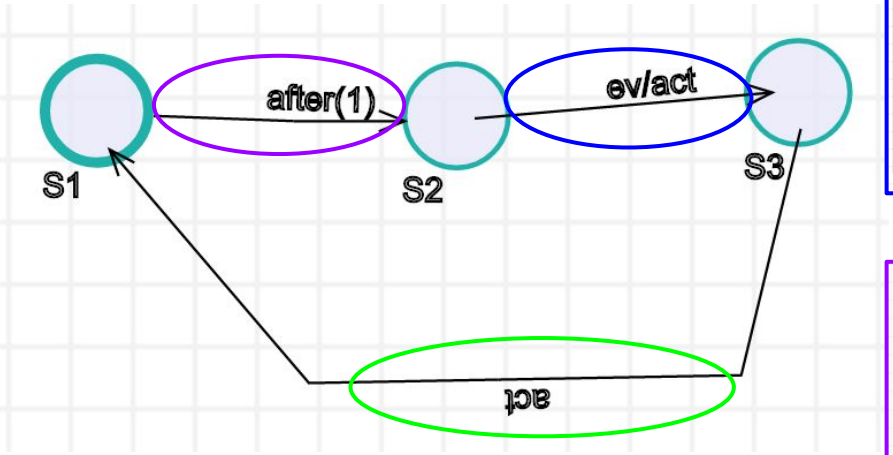


Boundaries

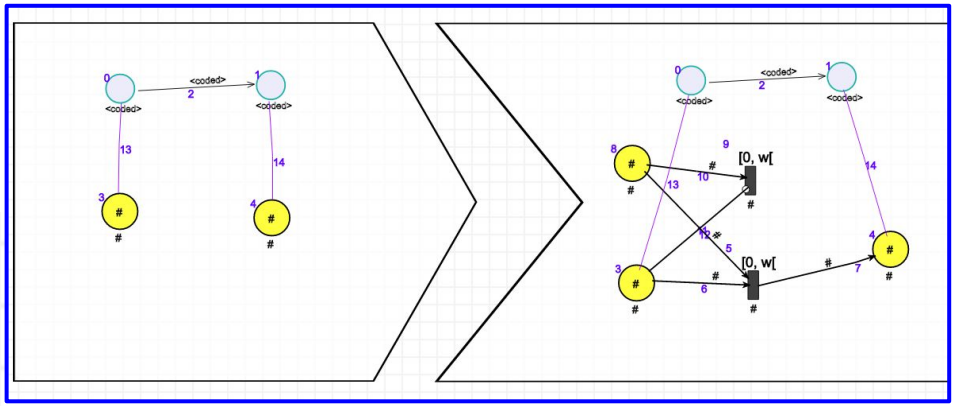

Transformation
entry/exit actions
History States
[Conditions]
Transitions
Time
Events
[IN STATE Conditions]
Composite States
Orthogonal Components

Transformation by examples

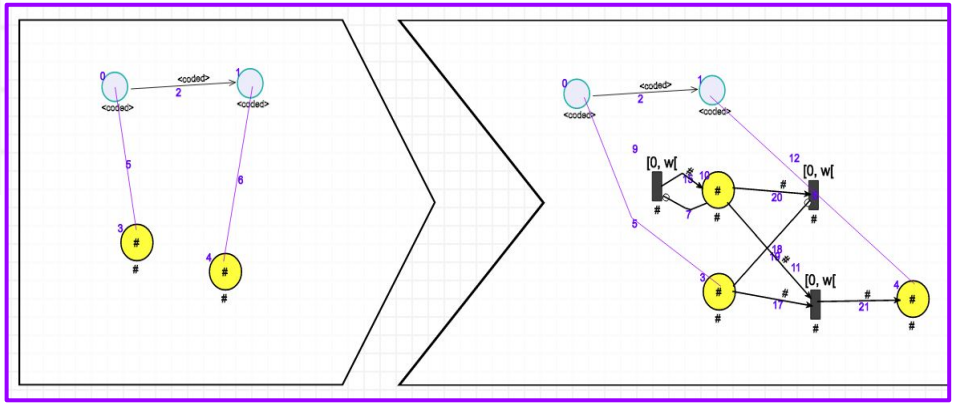
Transitions



Event Transition

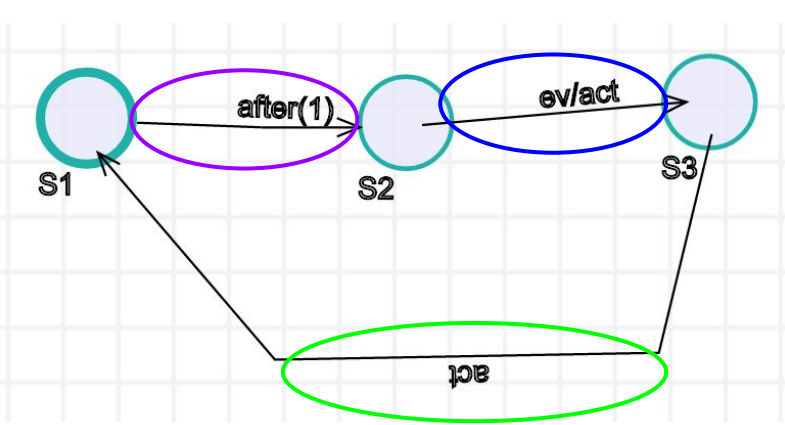


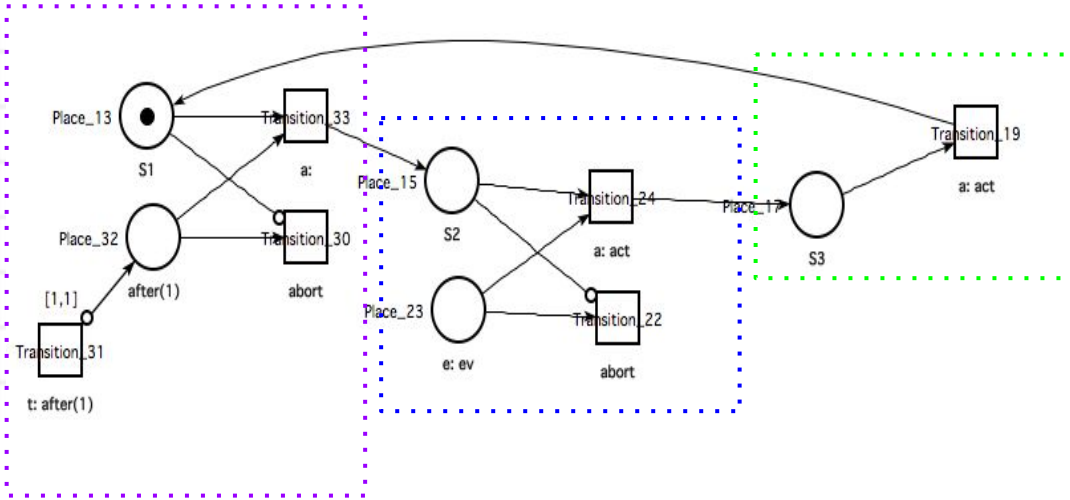
Timed Transition



Transformation by examples

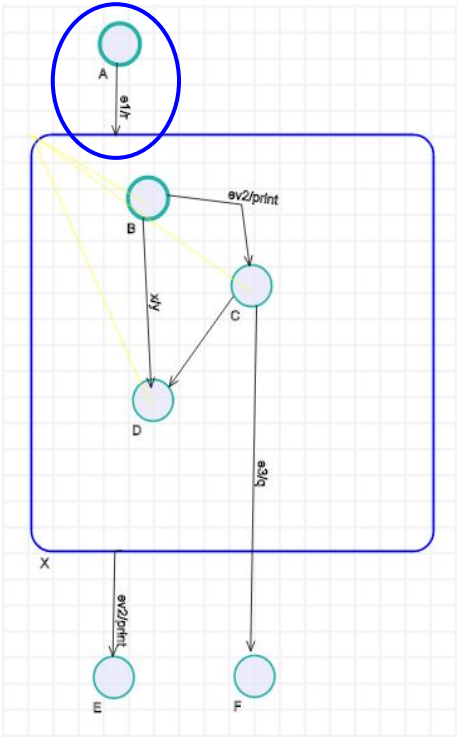
Transitions





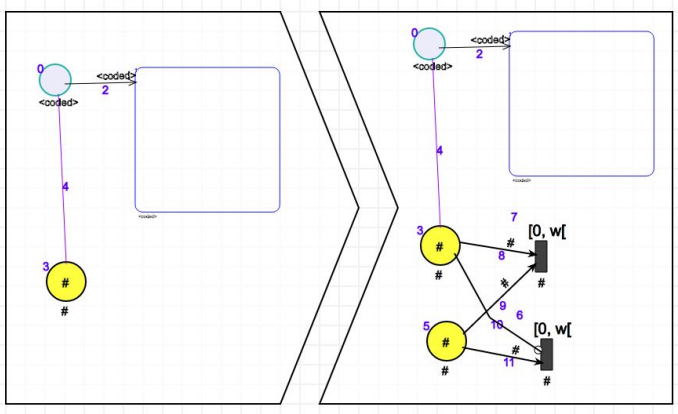
Transformation by examples

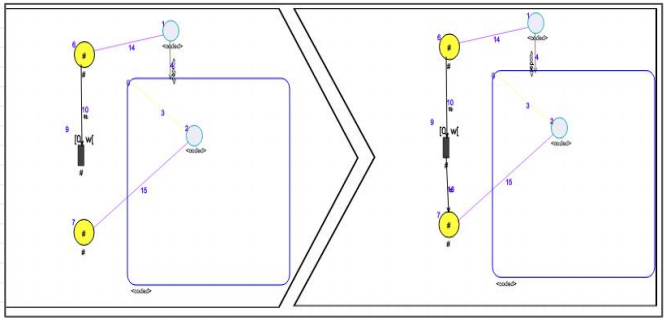
Composite States



Two steps approach:

1. Initialization
2. Connections
   a. Composite state -> Initial State
   b. Orthogonal Component ->Initial State of every orthogonal component



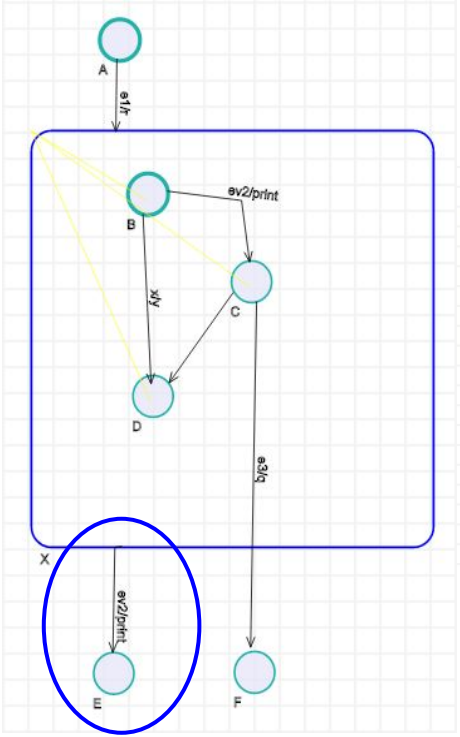


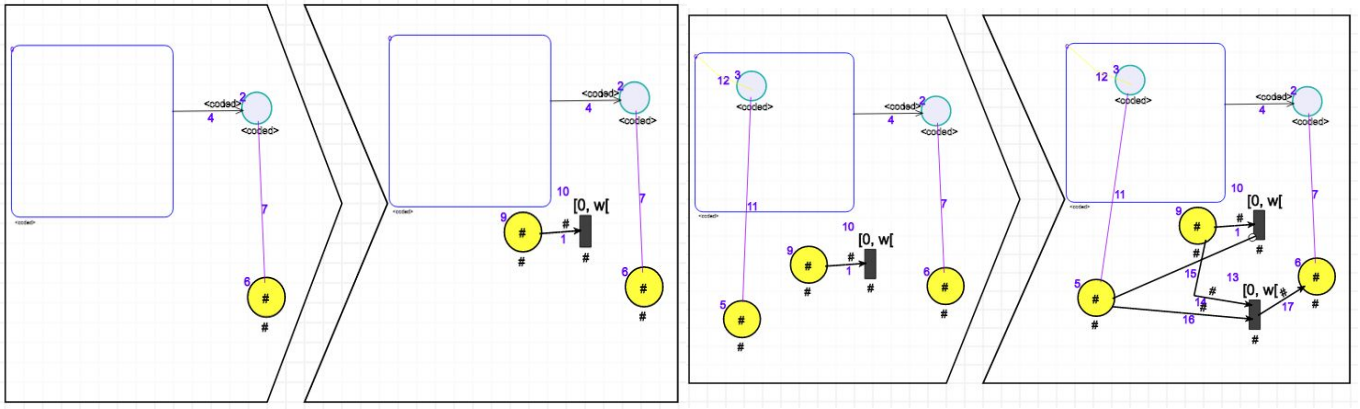
Transformation by examples

Composite States

Two steps approach:

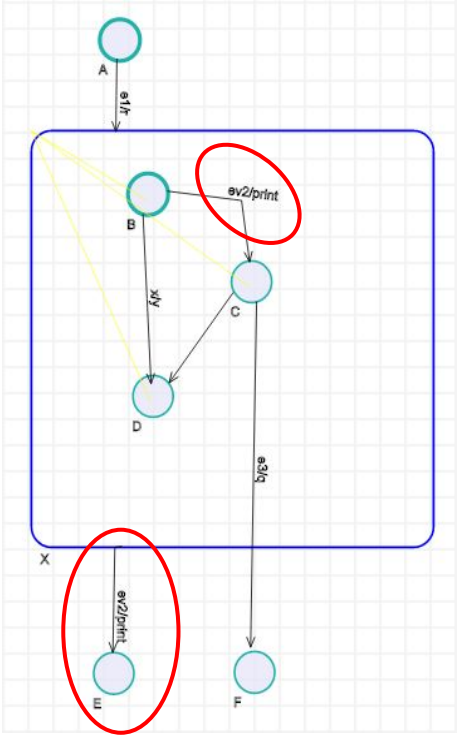
1. Initialization
2. Connection from every sub-state to the composite state





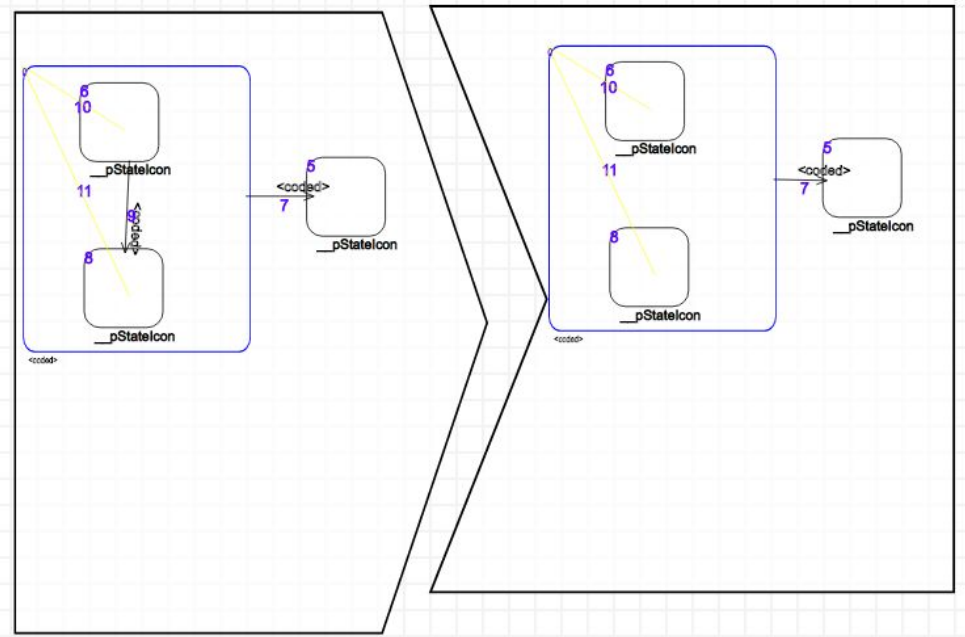
Transformation by examples

Non determinism



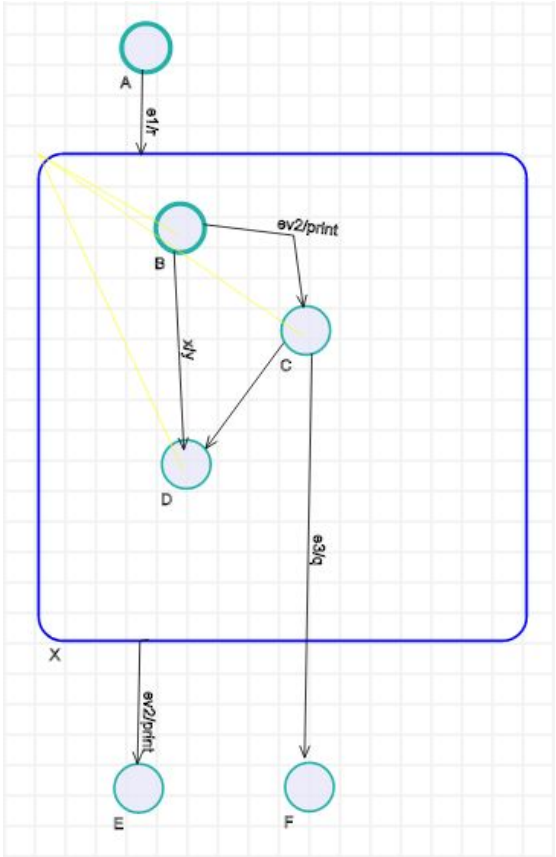
A choice is necessary:

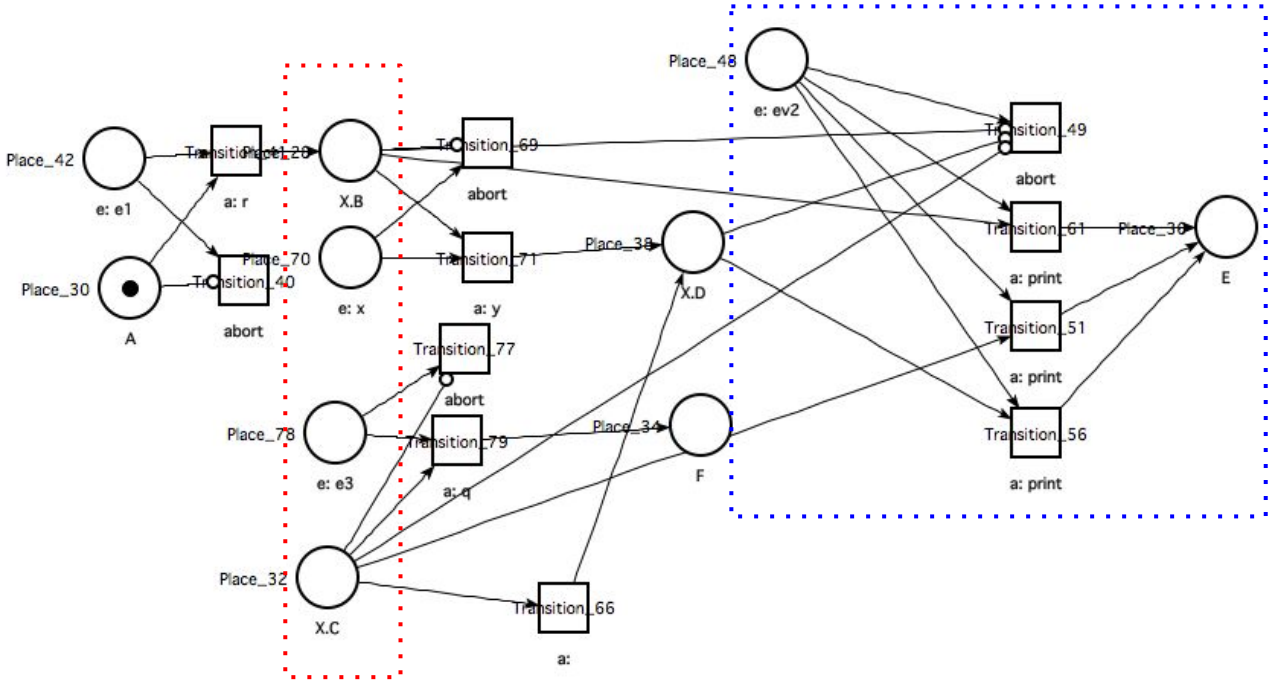
- StateMate approach (outer transitions)



D. Harel, A. Naamad, The statemate semantics of statecharts, ACM Trans. Softw. Eng. Methodol. 5 (4) (1996) 293–333

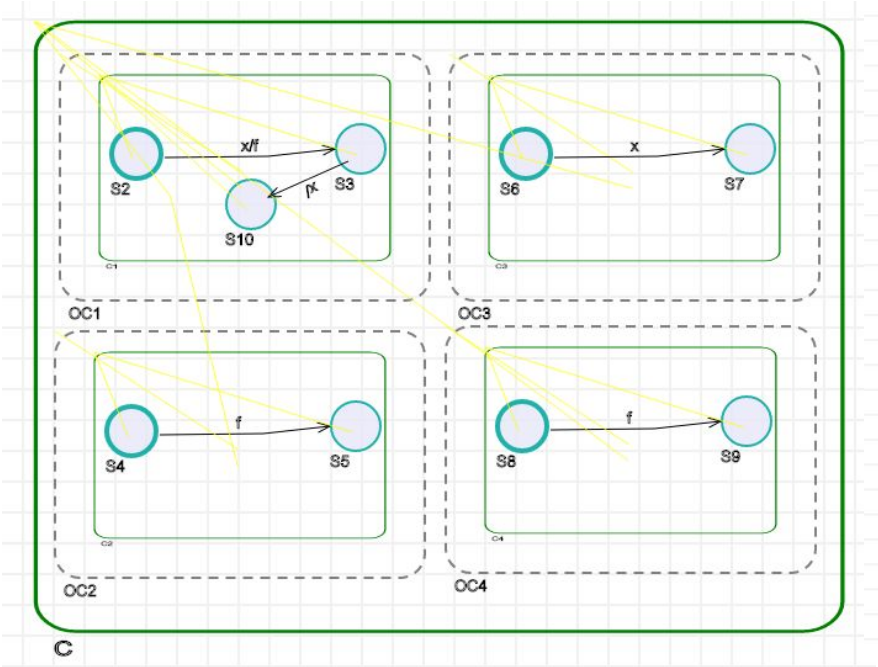
Transformation by examples

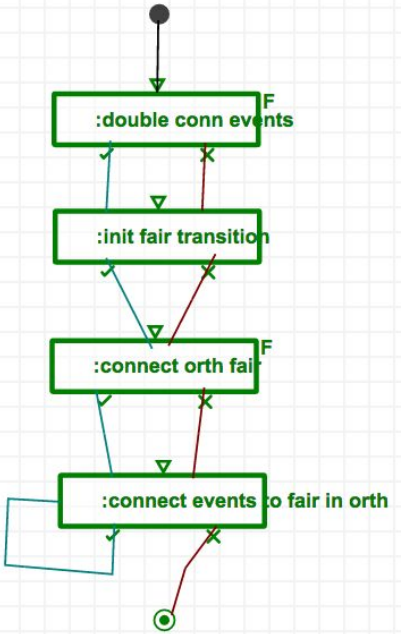




Transformation by examples

Orthogonal Components and Broadcasting



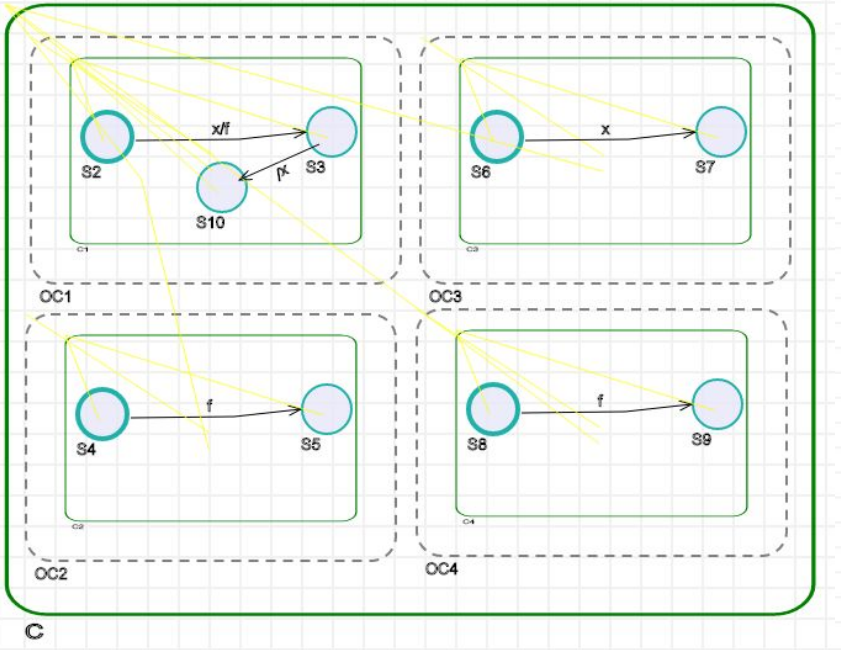


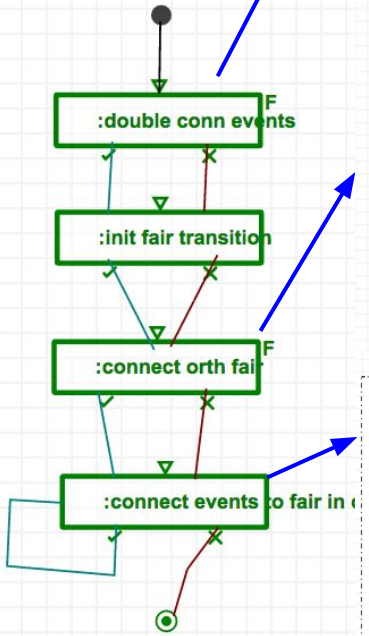
Note:

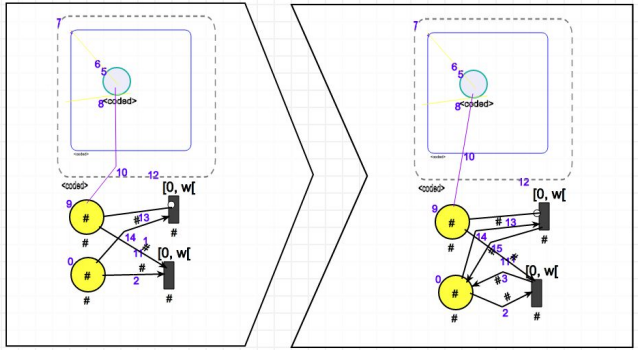
- every transition have the possibility to fire once in a time step;
- the events are consumed at the end of the time step.

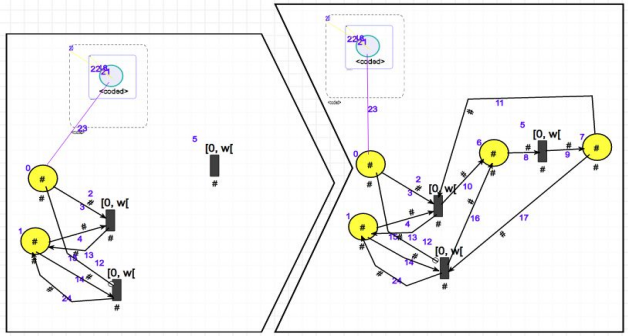
Transformation by examples

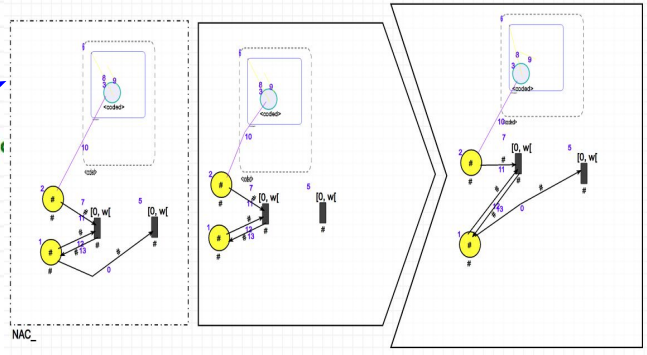
Orthogonal Components and Broadcasting





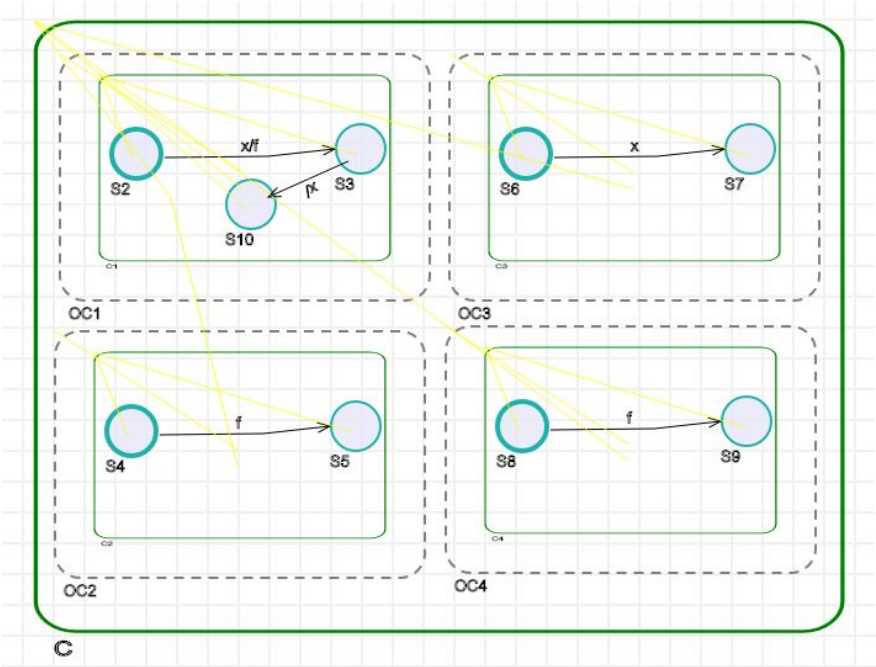


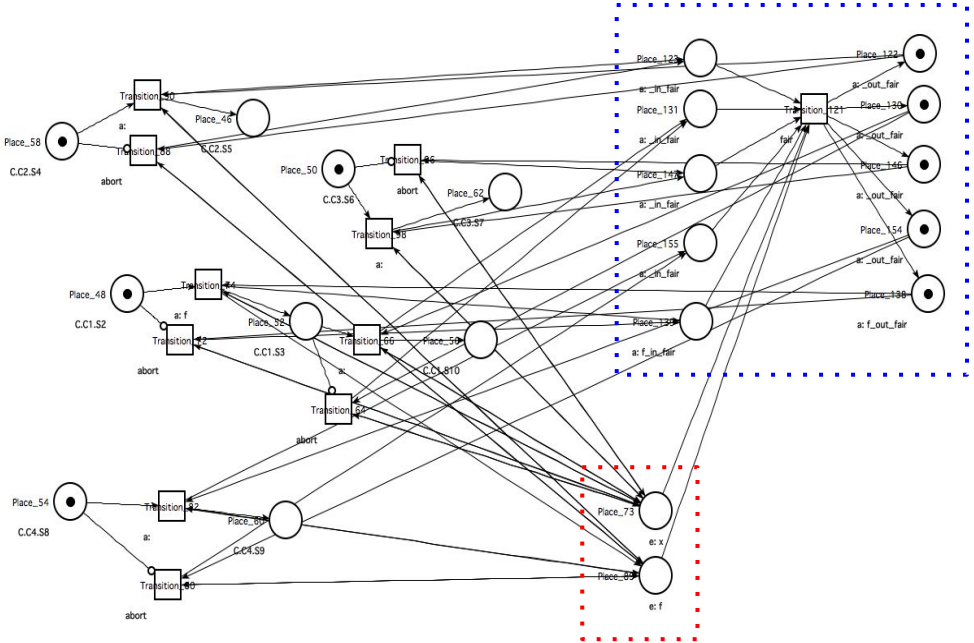




Transformation by examples

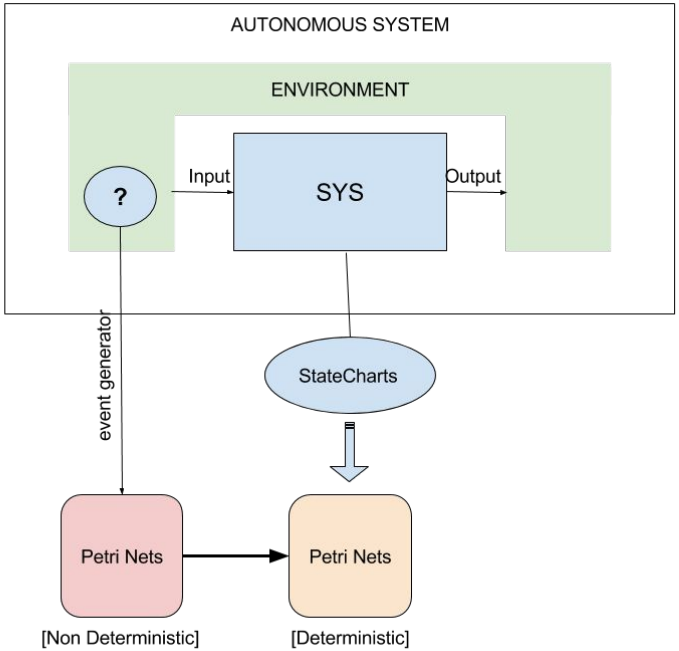
Orthogonal Components and Broadcasting



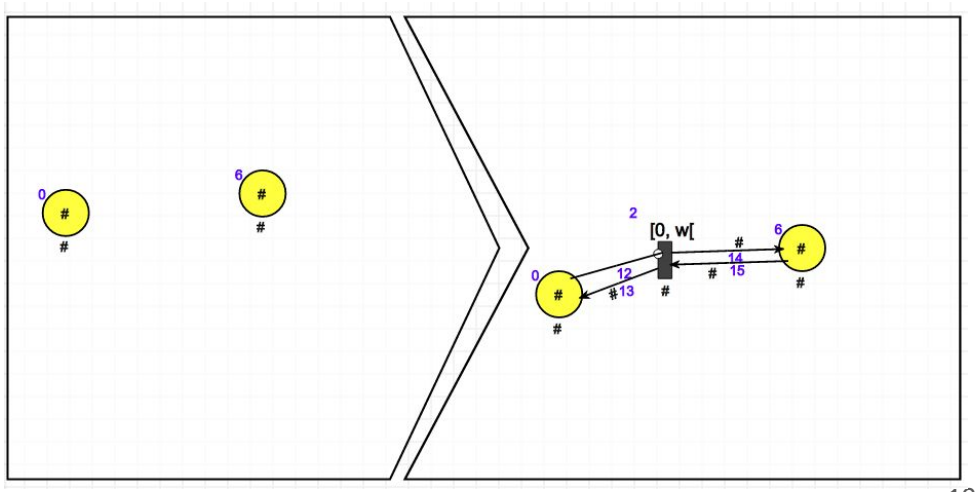


Transformation by examples

Somenthing is missing...Events Generator



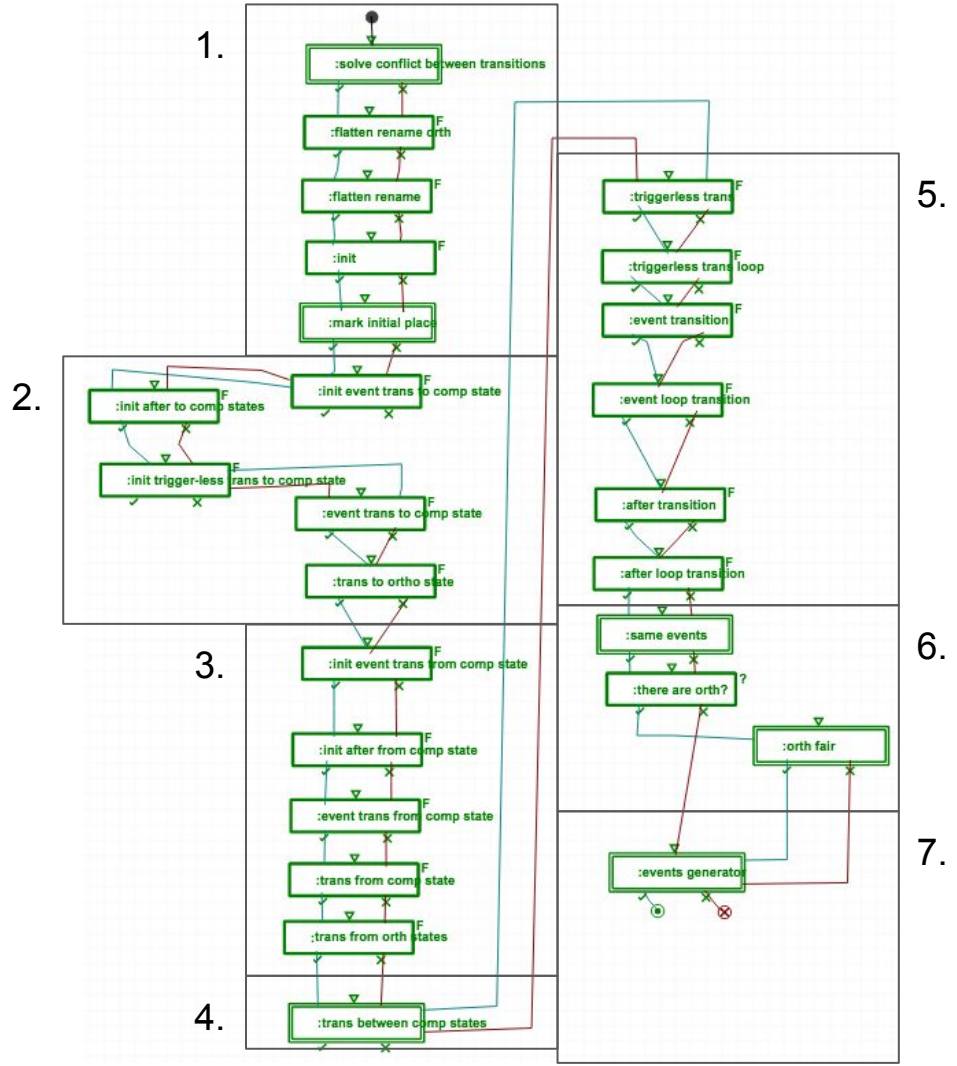
- We can simulate the interaction of the environment with the system (e.g user interfaces)
- We are able to construct the reachability graph (analyze the correctness of the transformation)



Put all together

Transformation Schedule:

1. Initialization
2. Handle transition to Composite States
3. Handle transition from Composite States
4. Handle transitions between Composite States
5. Handle transitions between Simple States
6. Handle events and broadcasting inside Orthogonal Components
7. Events Generator



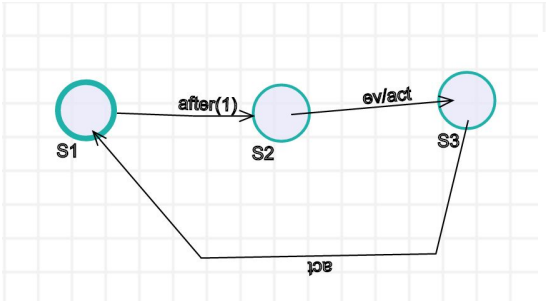
Export to TINA

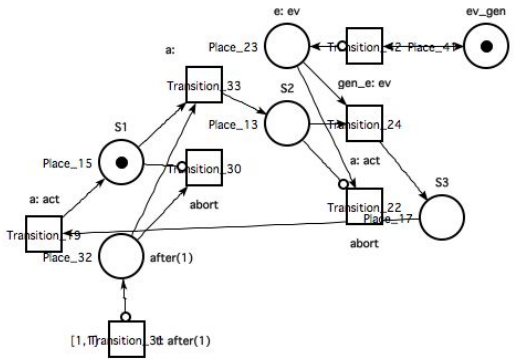
- Export in metaDepth:
  - TPN Abstract Syntax
  - TPN Model
- Convert in .tpn format using an egl script
- Import in TINA toolbox
- Assess goodness of the transformation:
  - comparison between manual and automatic nets
  - reachability graph

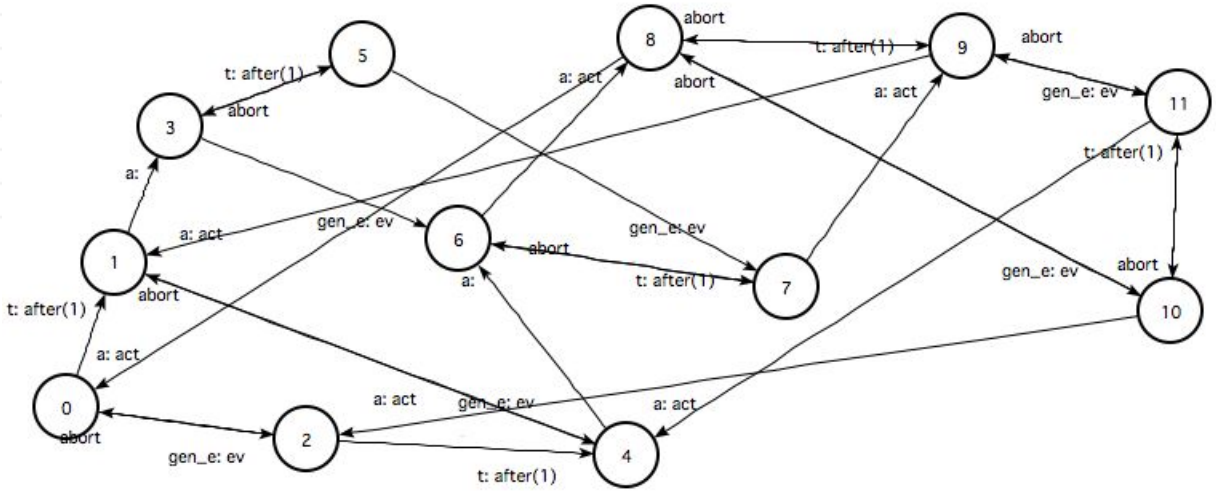
```
net sc_pn
tr Transition_54 : {a:} [0,w[ Place_34 -> Place_30
tr Transition_65 : {abort} [0,w[ Place_66 Place_34?-1 ->
tr Transition_76 : {gen_e: e3} [0,w[ Place_75 Place_66?-1 -> Place_75 Place_66
tr Transition_44 : {a:} [0,w[ Place_26 -> Place_28
tr Transition_67 : {a: q} [0,w[ Place_34 Place_66 -> Place_32
tr Transition_57 : {abort} [0,w[ Place_58 Place_26?-1 ->
tr Transition_37 : {abort} [0,w[ Place_58 Place_26?-1 Place_34?-1 Place_30?-1 ->
tr Transition_59 : {a:} [0,w[ Place_58 Place_26 -> Place_34
tr Transition_49 : {a:} [0,w[ Place_30 -> Place_28
tr Transition_39 : {a:} [0,w[ Place_34 Place_58 -> Place_28
tr Transition_81 : {gen_e: x} [0,w[ Place_75 Place_58?-1 -> Place_58 Place_75
pl Place_32 : {F} (0)
pl Place_75 : {ev_gen} (1)
pl Place_30 : {X.D} (0)
pl Place_28 : {E} (0)
pl Place_26 : {X.B} (0)
pl Place_58 : {e: x} (0)
pl Place_34 : {X.C} (0)
pl Place_66 : {e: e3} (0)
```

Export to TINA

Assess goodness of the transformation: Reachability Graph







http://projects.laas.fr/tina/

Future Work

- Extend boundaries:
  - support of history states, conditions, …
- Generalize rules using abstract states
- Improve performance

Thank you for the attention

Questions?