Asserting the Correctness of Translations

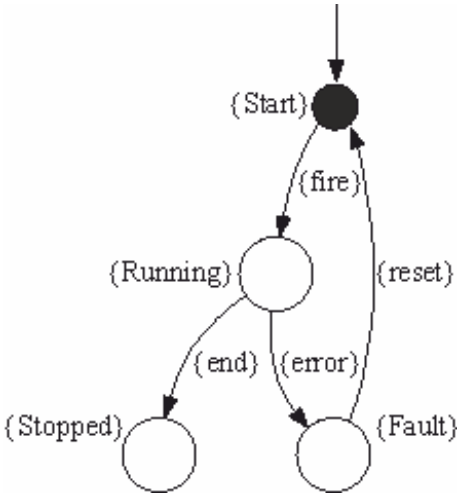
Bruno F. Barroca
(bruno.barroca@di.fct.unl.pt)

Vasco M. Amaral
(vasco.amaral@di.fct.unl.pt)

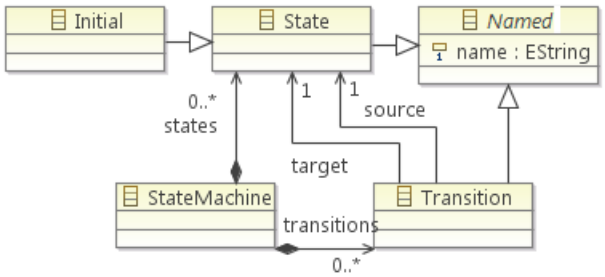
CITI-FCT/UNL

Motivating Example: StateMachines to Petri Nets

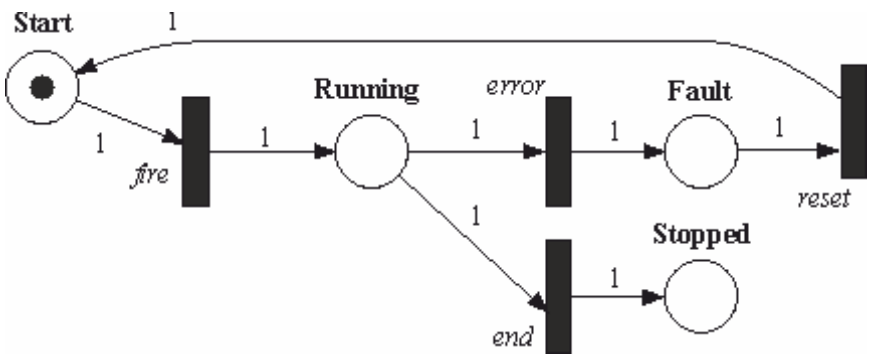
M_{MM}



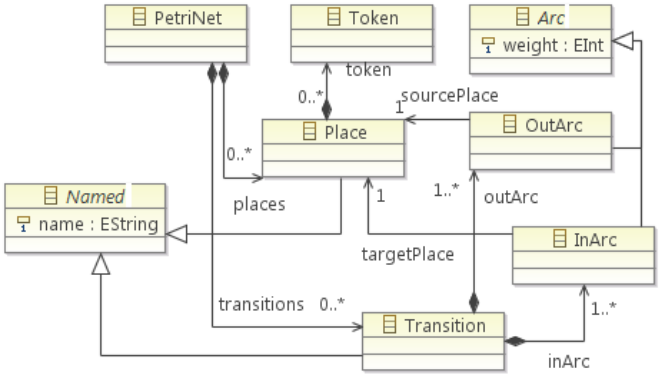
MM



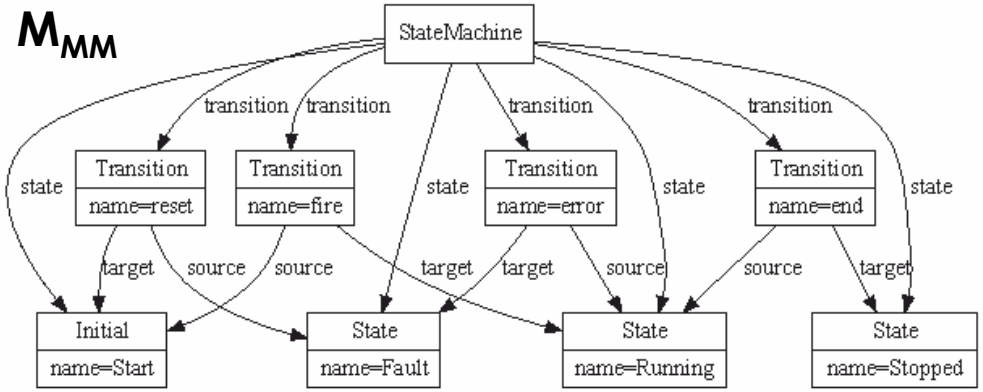
$M'_{MM'}$

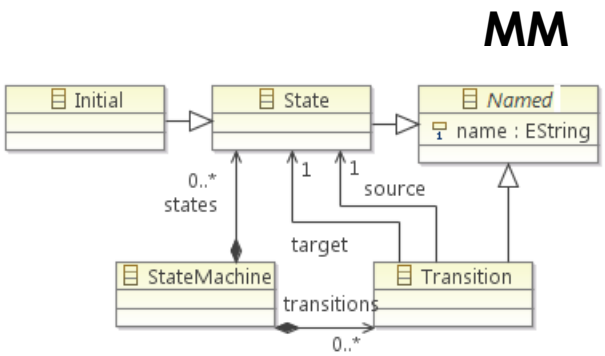


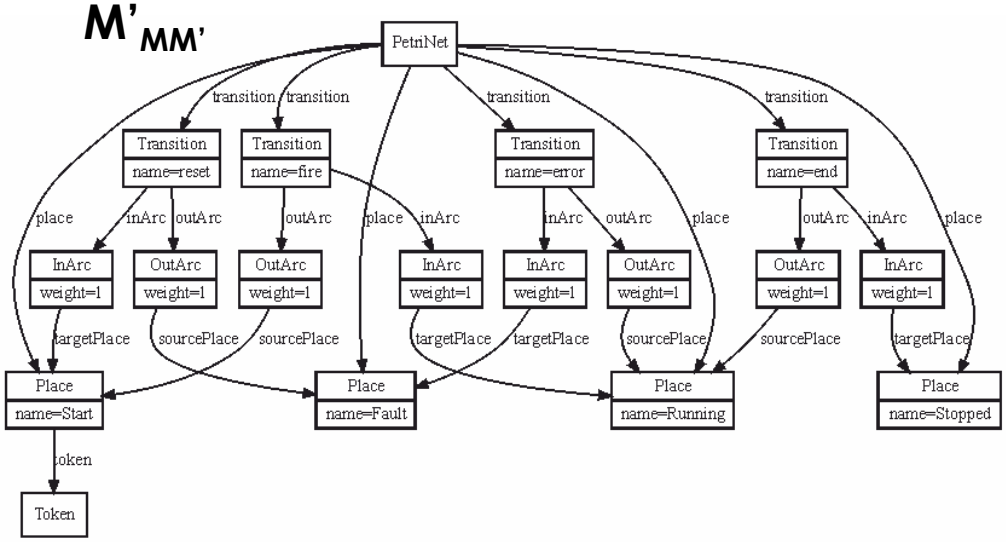
MM'

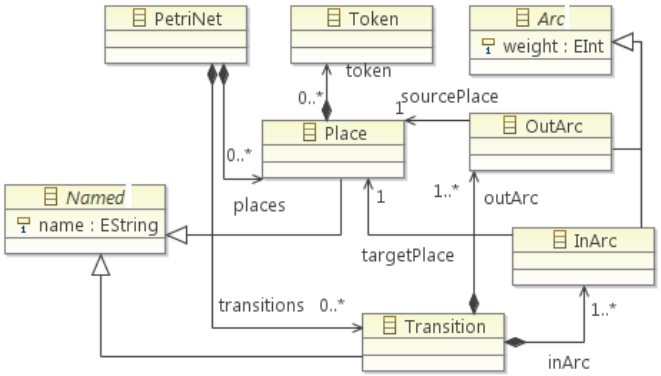


Motivating Example: StateMachines to Petri Nets









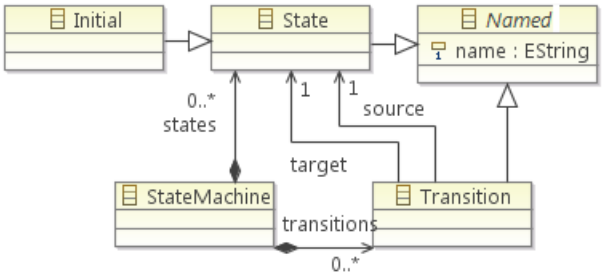
Motivating Example: StateMachine Semantics (SOS)

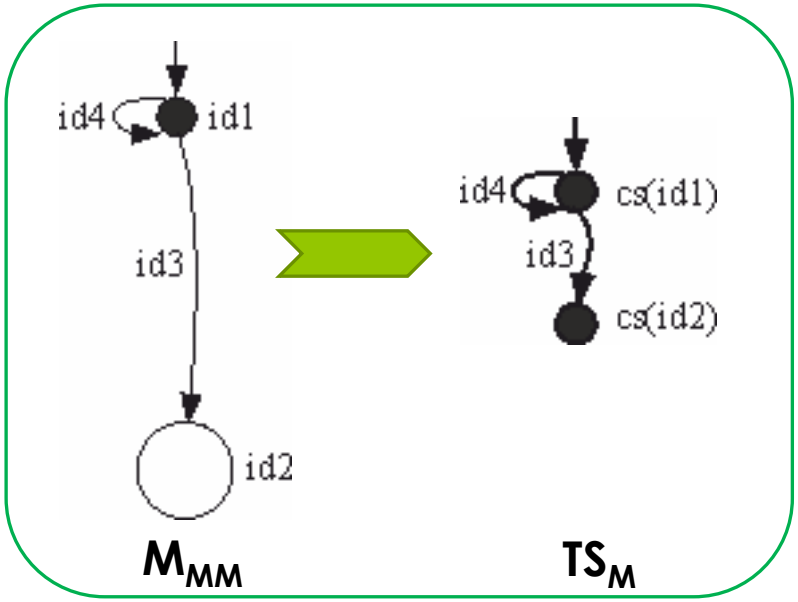
SEM$_{MM}$

$$Initial_s \Longrightarrow cs(Initial_s)$$

$$\frac{(Transition_s \xrightarrow{source} State_s) \wedge (Transition_s \xrightarrow{target} State'_s)}{cs(State_s) \xrightarrow{transition} cs(State'_s) \in TS_s}$$

MM





Motivating Example: StateMachine Semantics (SOS): Implementation

SEM$_{MM}$

$$Initial_s \implies cs(Initial_s)$$

$$\frac{(Transition_s \xrightarrow{source} State_s) \wedge (Transition_s \xrightarrow{target} State'_s)}{cs(State_s) \xrightarrow{transition} cs(State'_s) \in TS_s}$$

```
1
2 :- [smmodel].
3
4 eval(nil,R):-
5         model(States,Arcs), member(state(X,true),States),
6         eval((nil,nil,X),R).
7
8 eval((P,K,X),((P,K,X),T,Y)):-
9         model(States,Arcs),
10                member(state(X,_),States),
11                member(transition(T,X,Y),Arcs).
12
13 evalR(Z):- evalR(nil,Z).
14 evalR(State,NewState):- eval(State,NewState).
15 evalR(State,NewState):- eval(State,R), evalR(R,NewState).
```

Motivating Example: StateMachine Semantics (SOS): Implementation

SEM$_{MM}$

$$Initial_s \implies cs(Initial_s)$$

$$\frac{(Transition_s \xrightarrow{source} State_s) \wedge (Transition_s \xrightarrow{target} State'_s)}{cs(State_s) \xrightarrow{transition} cs(State'_s) \in TS_s}$$

```
model(
        [
                state(id1,true),
                state(id2,false)
        ],
        [
                transition(id4,id1,id1),
                transition(id3,id1,id2)
        ]
).
```

Motivating Example: StateMachine Semantics (SOS): Implementation

SEM$_{MM}$

$$Initial_s \implies cs(Initial_s)$$

$$\frac{(Transition_s \xrightarrow{source} State_s) \wedge (Transition_s \xrightarrow{target} State'_s)}{cs(State_s) \xrightarrow{transition} cs(State'_s) \in TS_s}$$

```
2 ?- evalR(X).
X = ((nil, nil, id1), id4, id1) ;
X = ((nil, nil, id1), id3, id2) ;
X = (((nil, nil, id1), id4, id1), id4, id1) ;
X = (((nil, nil, id1), id4, id1), id3, id2) ;
X = ((((nil, nil, id1), id4, id1), id4, id1), id4, id1) ;
X = ((((nil, nil, id1), id4, id1), id4, id1), id3, id2) ;
```

Motivating Example: PetriNet Semantics (SOS)

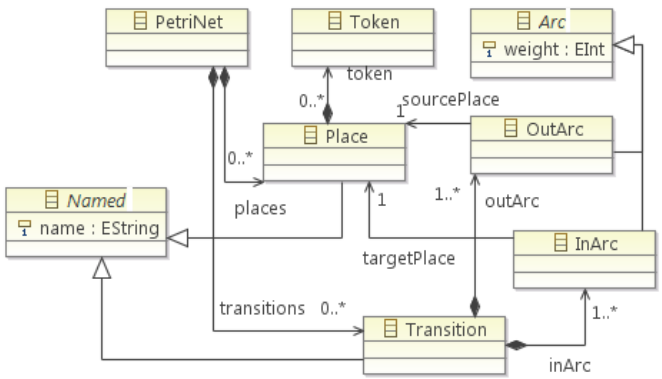
SEM$_{MM'}$

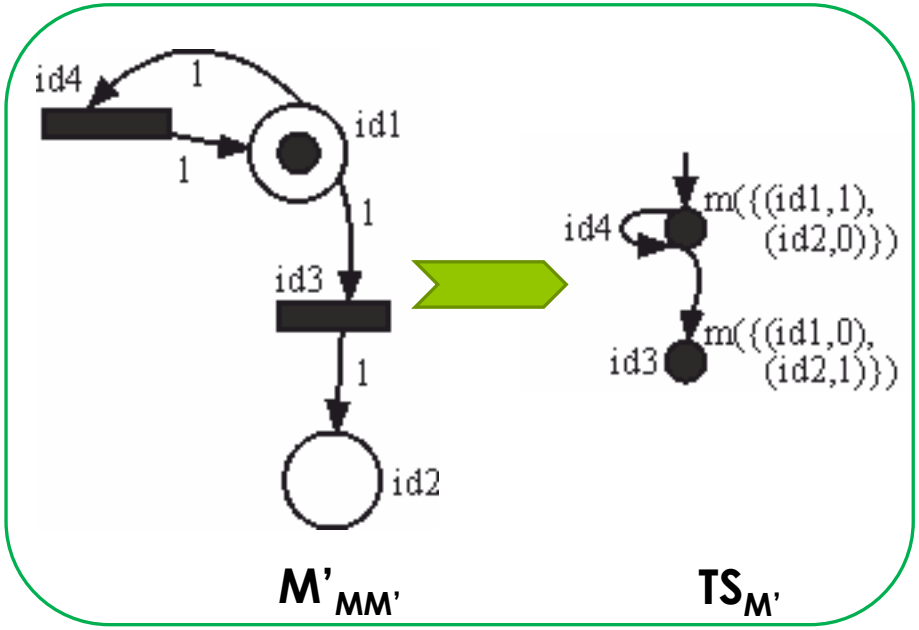
$$\frac{\begin{array}{c}\forall (Transition_p \xrightarrow{outArc} OutArc') : \\ (OutArc'_p \xrightarrow{sourcePlace} Place'_p) \wedge (Place'_p, T') \in M \implies OutArc'_p.weight \leq T'\end{array}}{m(M = \{(Place_p, T)\}) \xrightarrow{transition} m(\{(Place_p, T + InArc_p.weight - OutArc_p.weight)\}) \in TS_p}$$

$$InArc_p : (Transition_p \xrightarrow{inArc} InArc_p) \wedge (InArc_p \xrightarrow{targetPlace} Place_p)$$

$$OutArc_p : (Transition_p \xrightarrow{outArc} OutArc_p) \wedge (OutArc_p \xrightarrow{sourcePlace} Place_p)$$

MM'





Motivating Example: PetriNet Semantics (SOS): Implementation

SEM$_{MM'}$

$$\frac{\forall (Transition_p \xrightarrow{outArc} OutArc') : (OutArc'_p \xrightarrow{sourcePlace} Place'_p) \wedge (Place'_p, T') \in M \implies OutArc'_p.weight \leq T'}{m(M = \{(Place_p, T)\}) \xrightarrow{transition} m(\{(Place_p, T + InArc_p.weight - OutArc_p.weight)\}) \in TS_p}$$

$$InArc_p : (Transition_p \xrightarrow{inArc} InArc_p) \wedge (InArc_p \xrightarrow{targetPlace} Place_p)$$

$$OutArc_p : (Transition_p \xrightarrow{outArc} OutArc_p) \wedge (OutArc_p \xrightarrow{sourcePlace} Place_p)$$

```
:- [pnmodel].

transitions(Arcs,Z):-
                findall(Tout,member(outarc(_,_,Tout),Arcs),Zout),
                findall(Tin,member(inarc(Tin,_,_),Arcs),Zin),
                union(Zout,Zin,Z).

eval(Nodes,Arcs,R):-
        transitions(Arcs,Trans), member(T,Trans),
        evalprecond(Nodes,T,Arcs),
        evalposcond(Nodes,Arcs,T,R).

evalR(Z):- model(Nodes,Arcs), evalR(Nodes,Arcs,Z).
evalR(Nodes,Arcs,R):- eval(Nodes,Arcs,R).
evalR(Nodes,Arcs,R):- eval(Nodes,Arcs,NewNodes), evalR(NewNodes,Arcs,R).

evalprecond(_,_,[]).
evalprecond(Nodes,T,[inarc(_,_,_)|L]):- evalprecond(Nodes,T,L).
evalprecond(Nodes,T,[outarc(_,_,K)|L]):- T \= K, evalprecond(Nodes,T,L).
evalprecond(Nodes,T,[outarc(N,W,T)|L]):-
                member(place(N,Token),Nodes),
                Token >= W, evalprecond(Nodes,T,L).
```

Motivating Example: PetriNet Semantics (SOS): Implementation

SEM$_{MM'}$

$$\frac{\forall (Transition_p \xrightarrow{outArc} OutArc') : (OutArc'_p \xrightarrow{sourcePlace} Place'_p) \wedge (Place'_p, T') \in M \implies OutArc'_p.weight \leq T'}{m(M = \{(Place_p, T)\}) \xrightarrow{transition} m(\{(Place_p, T + InArc_p.weight - OutArc_p.weight)\}) \in TS_p}$$

$$InArc_p : (Transition_p \xrightarrow{inArc} InArc_p) \wedge (InArc_p \xrightarrow{targetPlace} Place_p)$$

$$OutArc_p : (Transition_p \xrightarrow{outArc} OutArc_p) \wedge (OutArc_p \xrightarrow{sourcePlace} Place_p)$$

```
model(
        [
                place(id1,1),
                place(id2,0)
        ],
        [
                outarc(id1,1,id4),inarc(id4,1,id1),
                outarc(id1,1,id3),inarc(id3,1,id2)
        ]
).
```

Motivating Example: PetriNet Semantics (SOS): Implementation

SEM$_{MM'}$

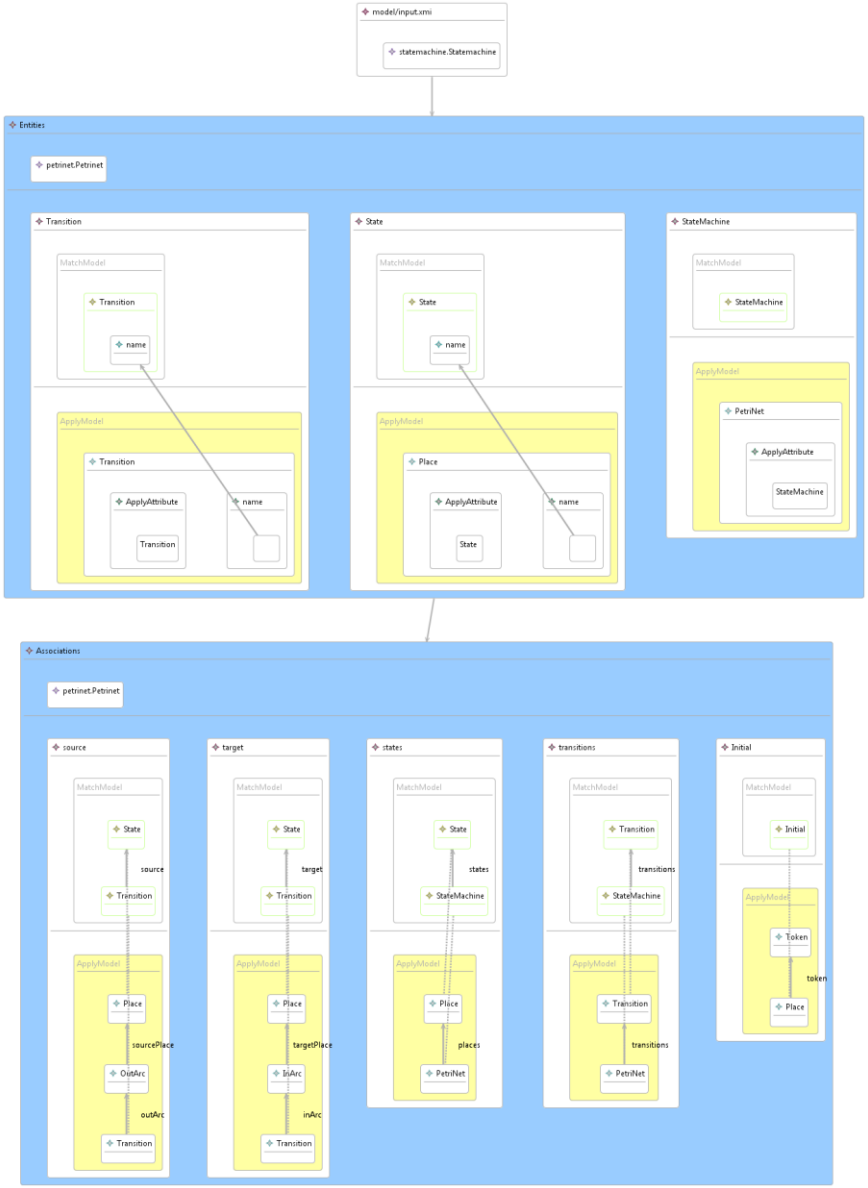
$$\frac{\begin{array}{c}\forall (Transition_p \xrightarrow{outArc} OutArc') : \\ (OutArc'_p \xrightarrow{sourcePlace} Place'_p) \wedge (Place'_p, T') \in M \implies OutArc'_p.weight \leq T'\end{array}}{m(M = \{(Place_p, T)\}) \xrightarrow{transition} m(\{(Place_p, T + InArc_p.weight - OutArc_p.weight)\}) \in TS_p}$$

$$InArc_p : (Transition_p \xrightarrow{inArc} InArc_p) \wedge (InArc_p \xrightarrow{targetPlace} Place_p)$$

$$OutArc_p : (Transition_p \xrightarrow{outArc} OutArc_p) \wedge (OutArc_p \xrightarrow{sourcePlace} Place_p)$$

```
1 ?- evalR(X).
X = [place(id1, 1), place(id2, 0)] ;
X = [place(id1, 0), place(id2, 1)] ;
X = [place(id1, 1), place(id2, 0)] ;
X = [place(id1, 0), place(id2, 1)] ;
X = [place(id1, 1), place(id2, 0)] ;
X = [place(id1, 0), place(id2, 1)] ;
X = [place(id1, 1), place(id2, 0)] ;
X = [place(id1, 0), place(id2, 1)] ;
```

Motivating Example: Translation Specification (in DSLTrans)



File Port: a State Machine

First Layer

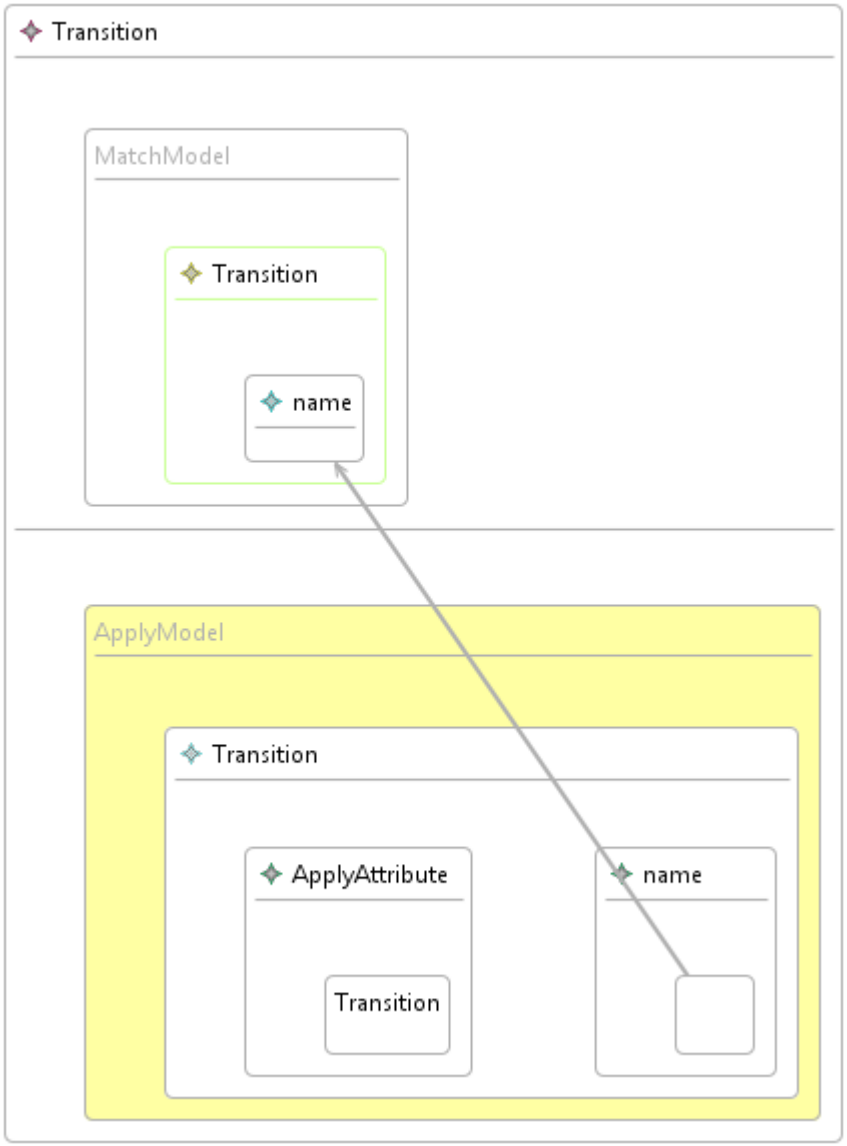
- State Machine Entities
- Petri Net Entities

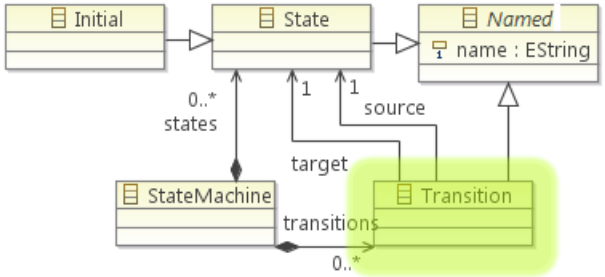
Second Layer

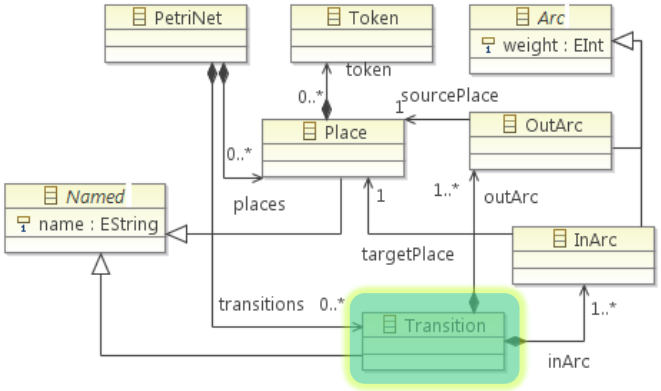
- State Machine Relations to Petri Net Relations

Output: a Petri Net

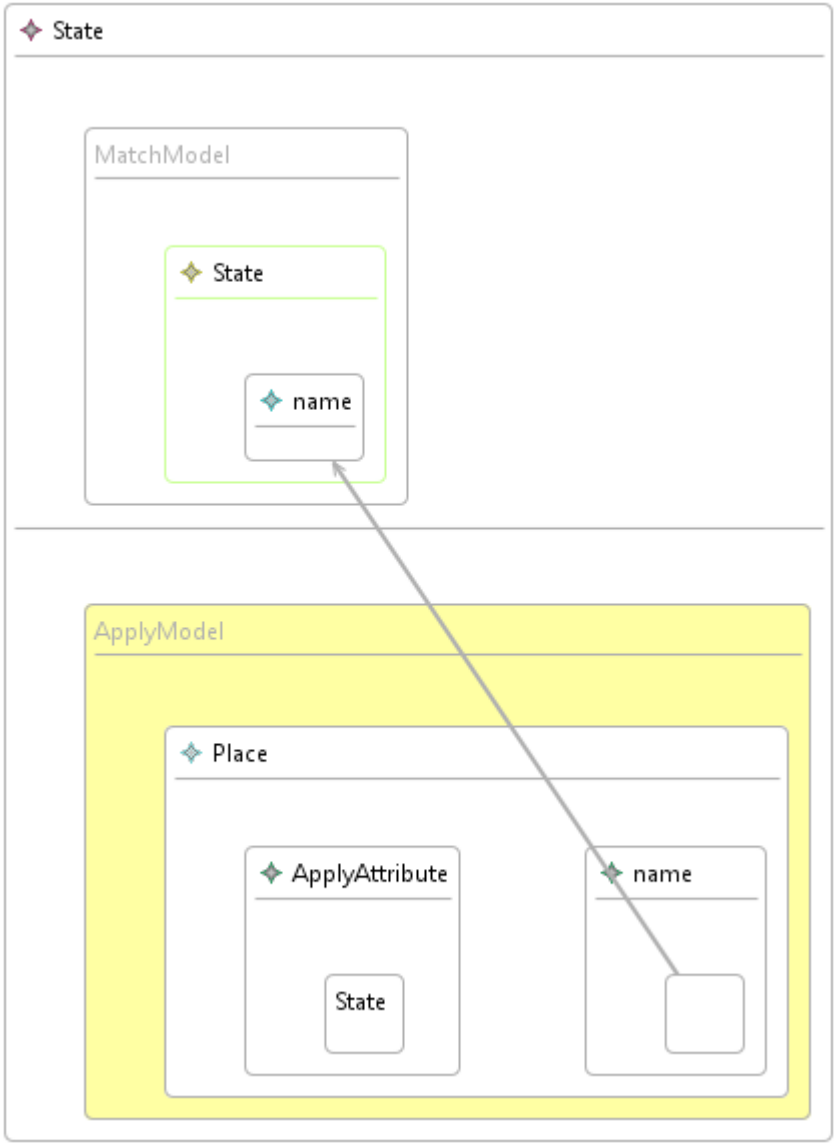
Motivating Example: Translation: First Layer: Transition Rule

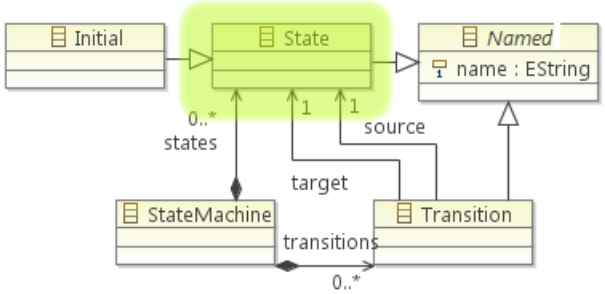


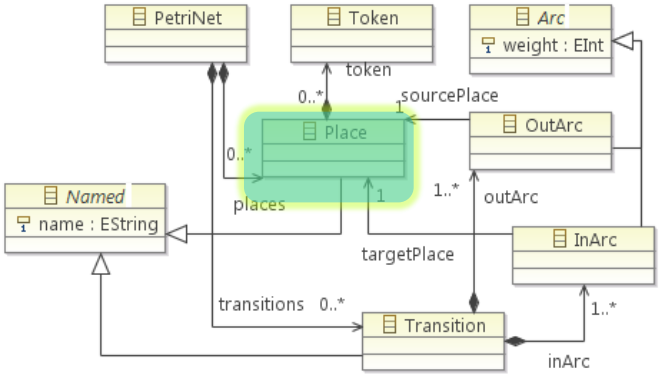




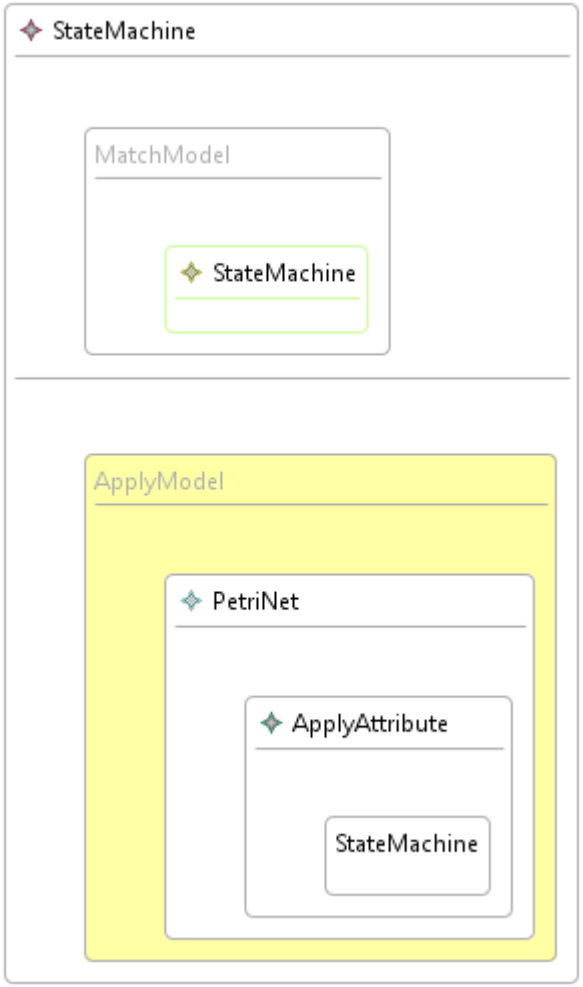
Motivating Example: Translation: First Layer: State Rule

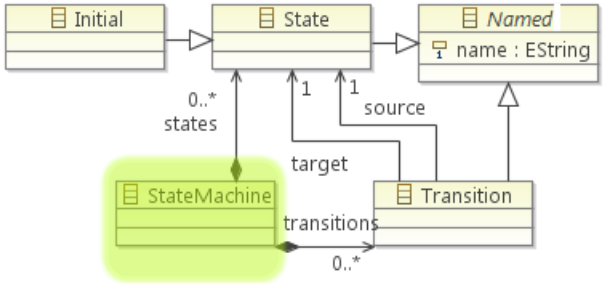


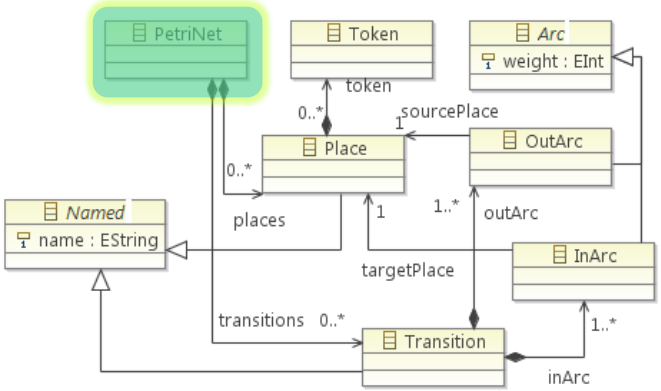




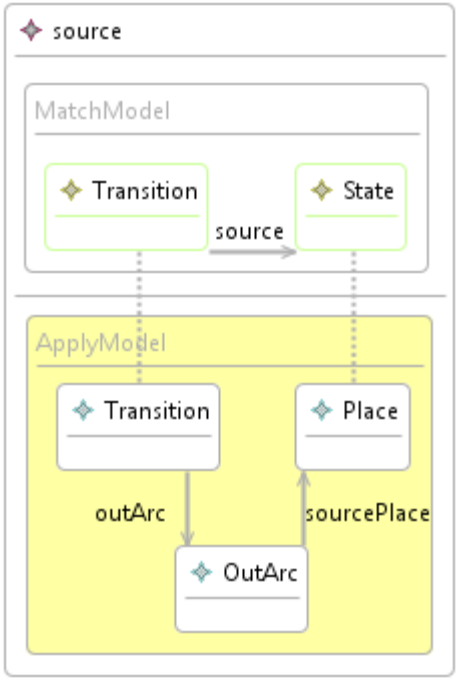
Motivating Example: Translation: First Layer: StateMachine Rule

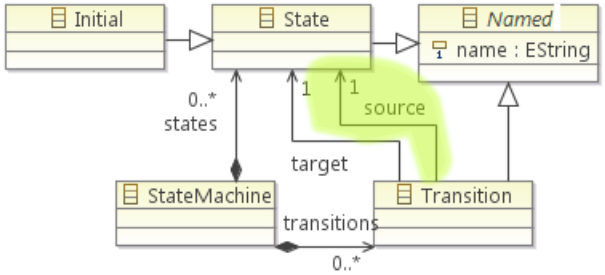


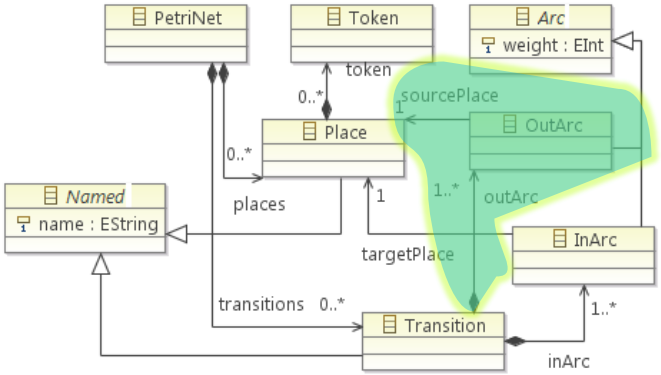




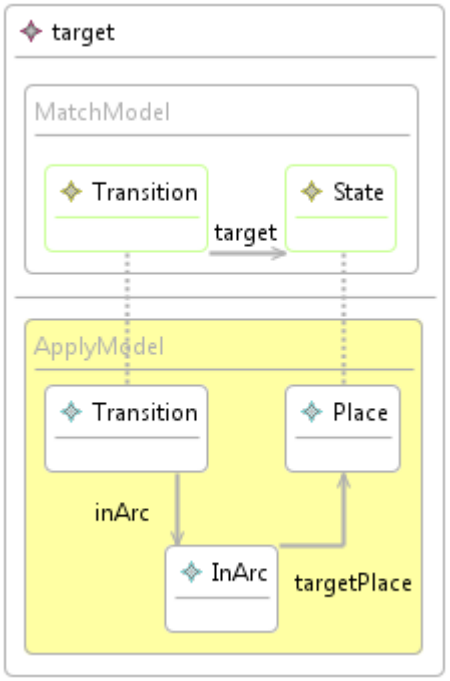
Motivating Example: Translation: Second Layer: source Rule

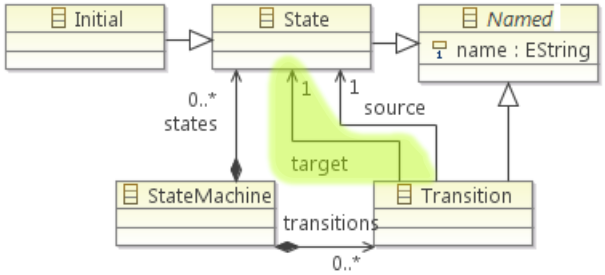


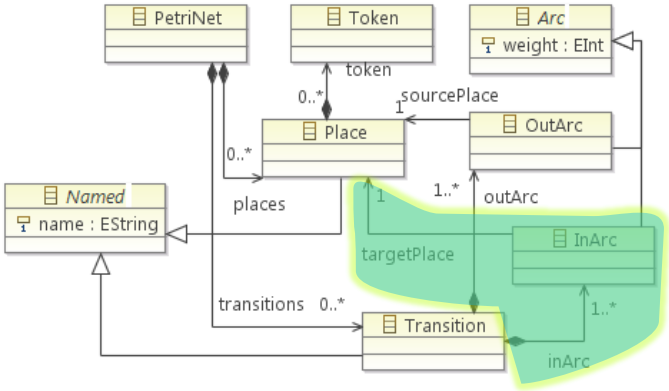




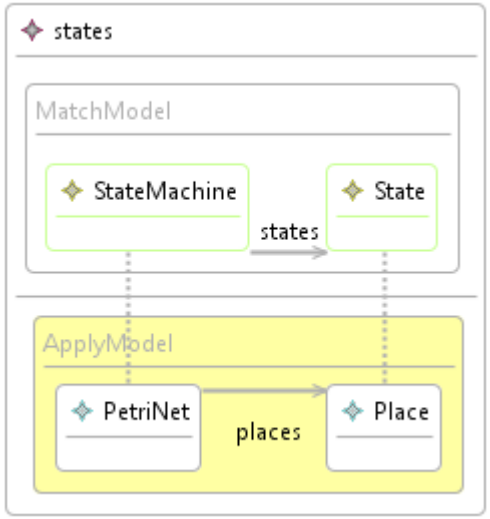
Motivating Example: Translation: Second Layer: target Rule

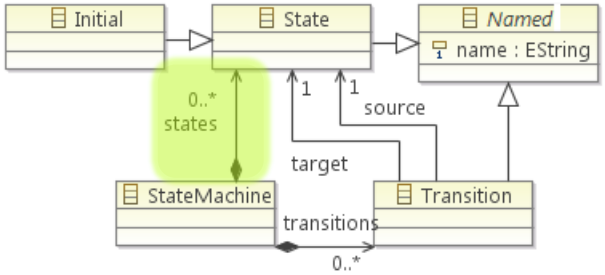


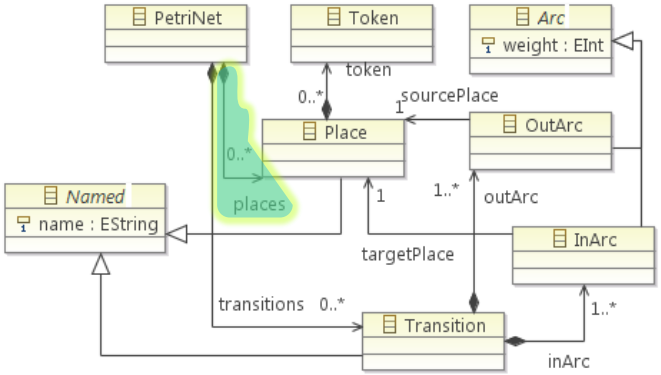




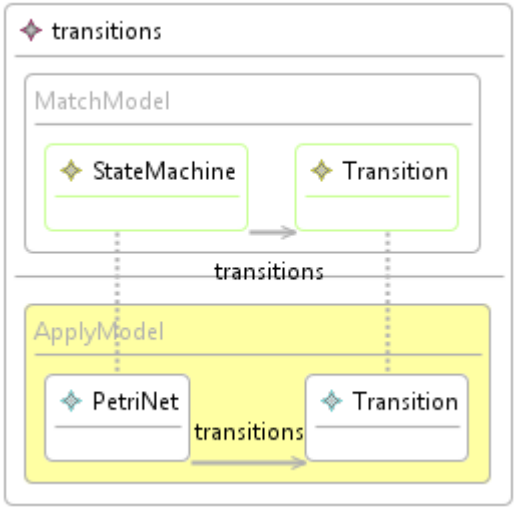
Motivating Example: Translation: Second Layer: states Rule

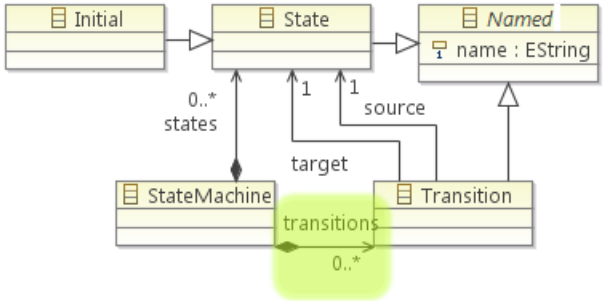


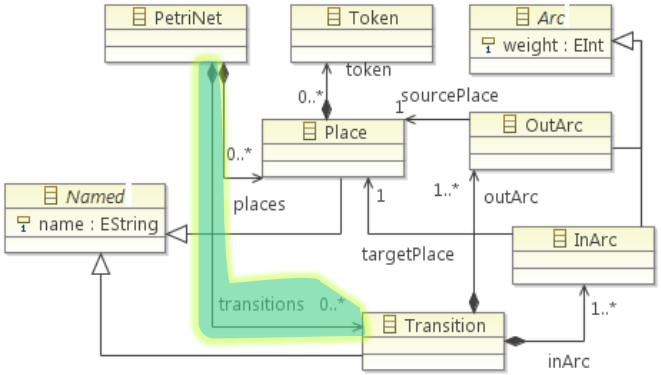




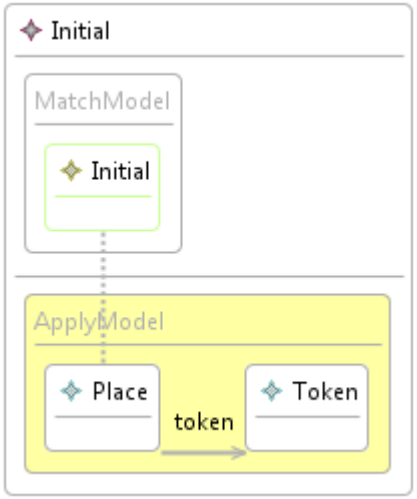
Motivating Example: Translation: Second Layer: transitions Rule

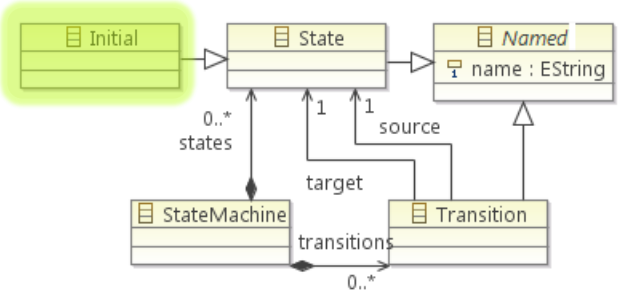


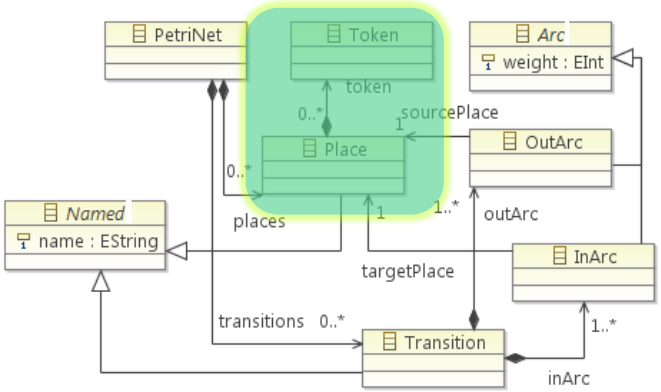




Motivating Example: Translation: Second Layer: Initial Rule



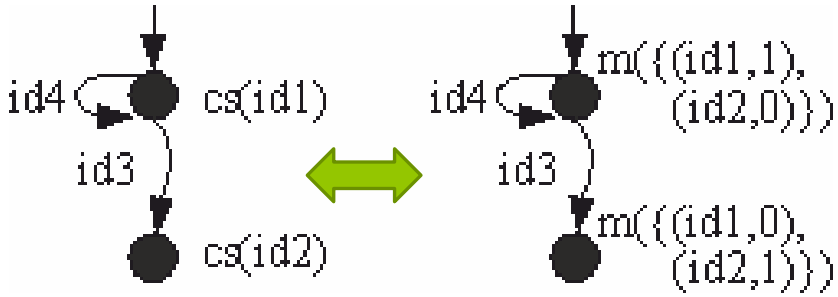




Hypothesis

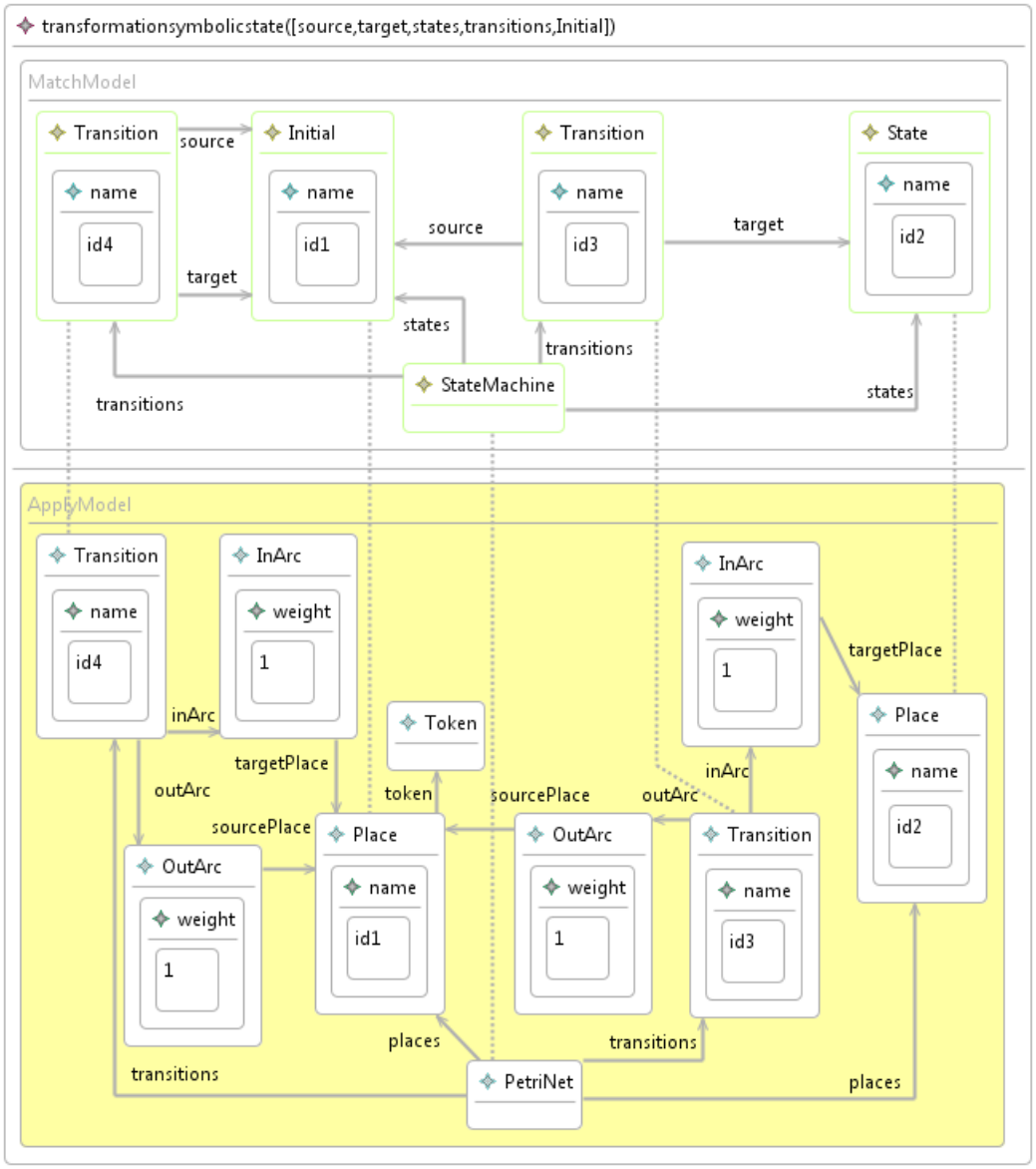
$$\begin{tikzcd}
M_{mm} \arrow[r, "translation_t"] \arrow[d, "Sem_{mm}"'] & M'_{mm'} \arrow[d, "Sem_{mm'}"] \\
TS_M \arrow[r, "equivalence_t"] & TS_{M'}
\end{tikzcd}$$

- M and M' can be infinite
- TS_M and $TS_{M'}$ can be infinite
- Semantic Equivalence
  - Bisimulation



  - How to Validate our translation?

Experiment: Translation: Analysis I



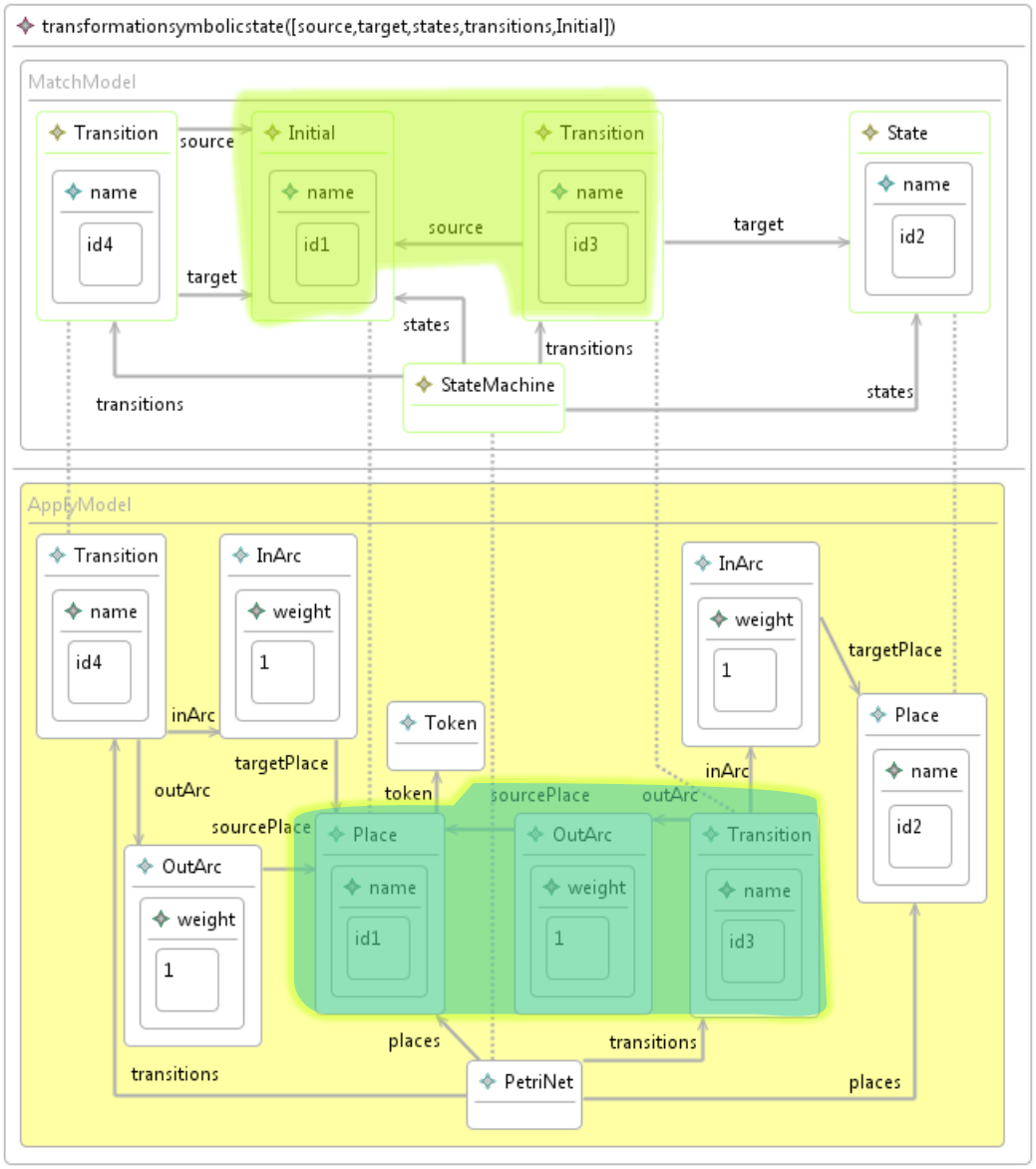
One result from the combination of the following translation rules:

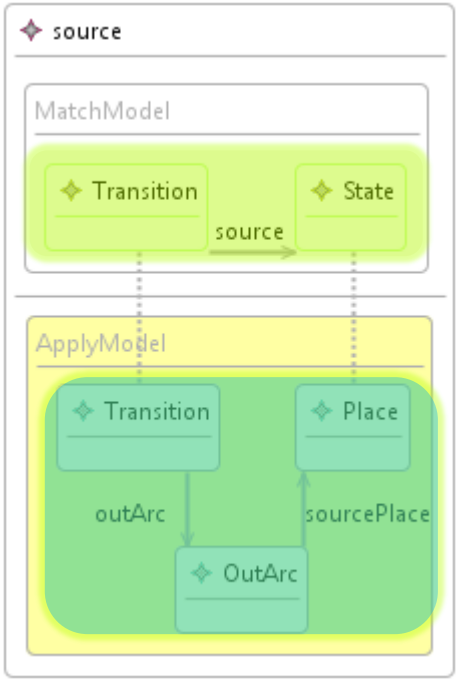
- source Rule
- target Rule
- states Rule
- transitions Rule

and

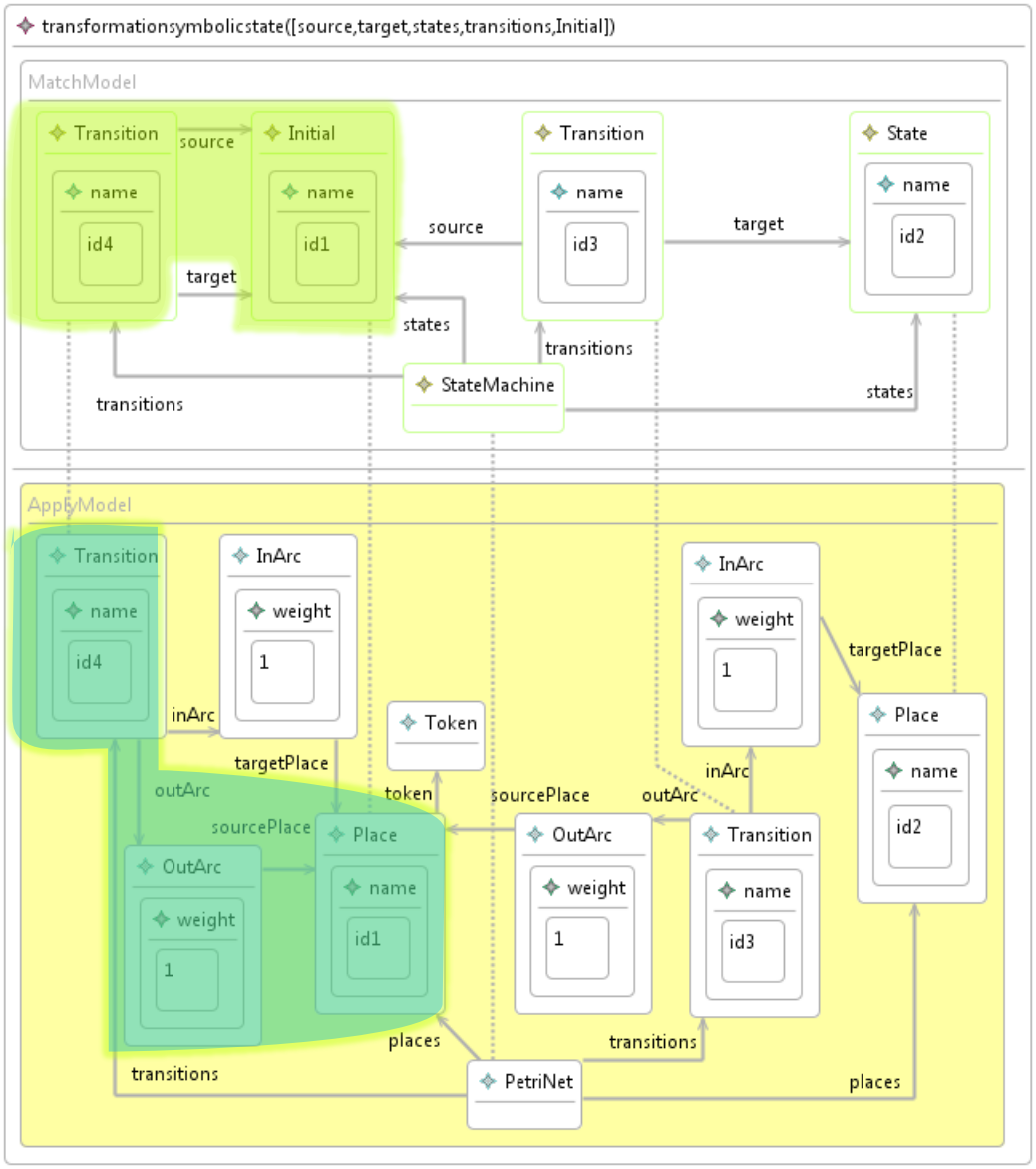
- Transition Rule
- State Rule
- StateMachine Rule
- Initial Rule

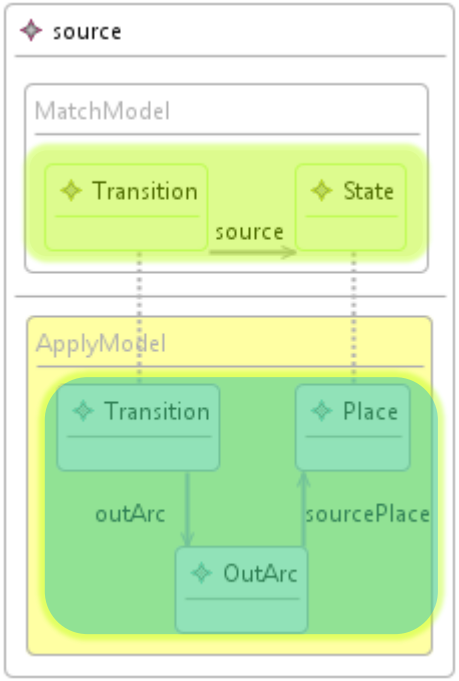
Experiment: Translation: Analysis I



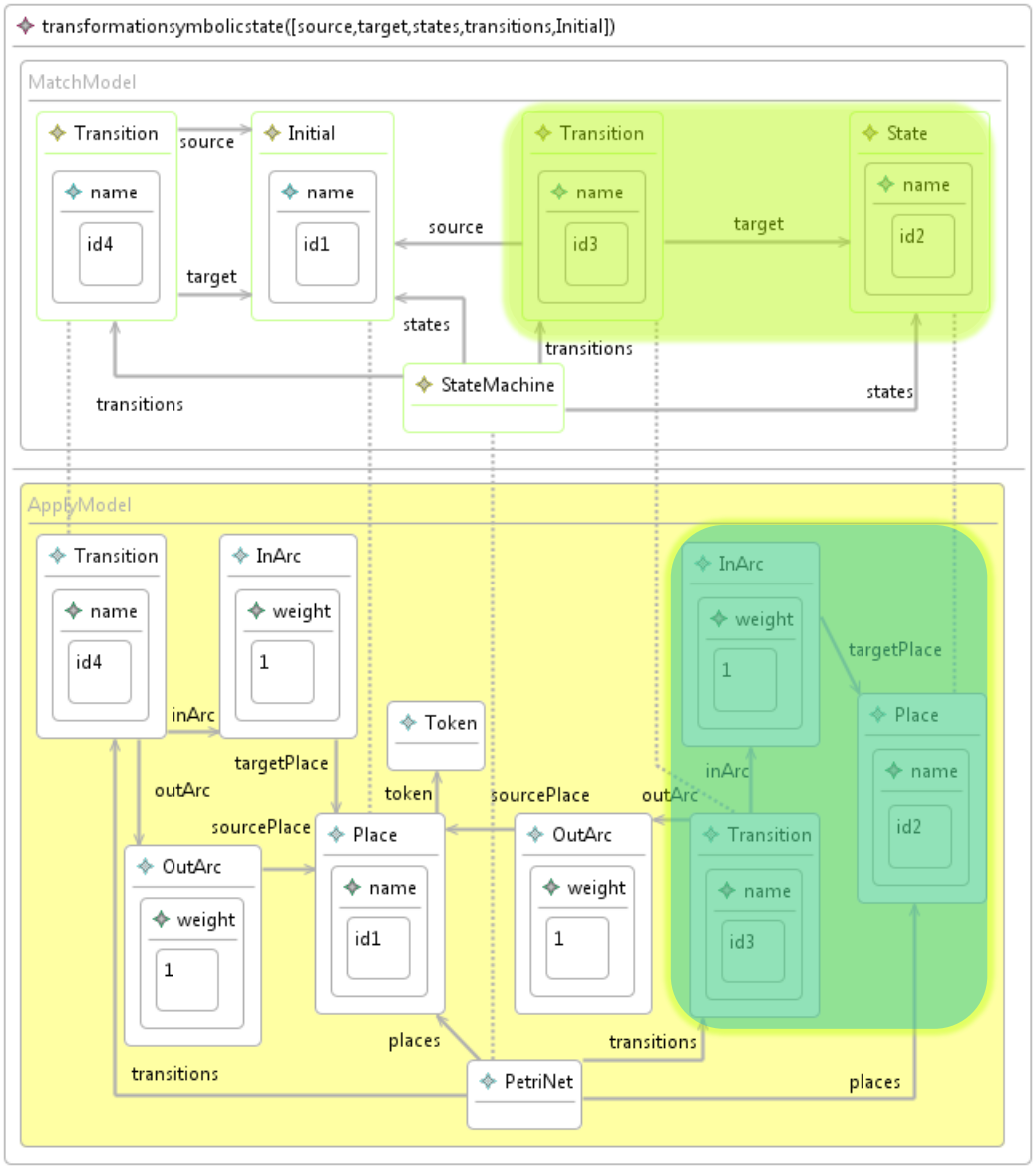


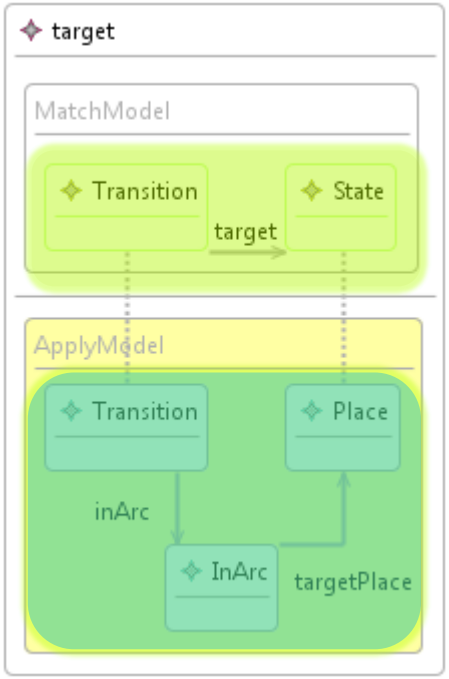
Experiment: Translation: Analysis I



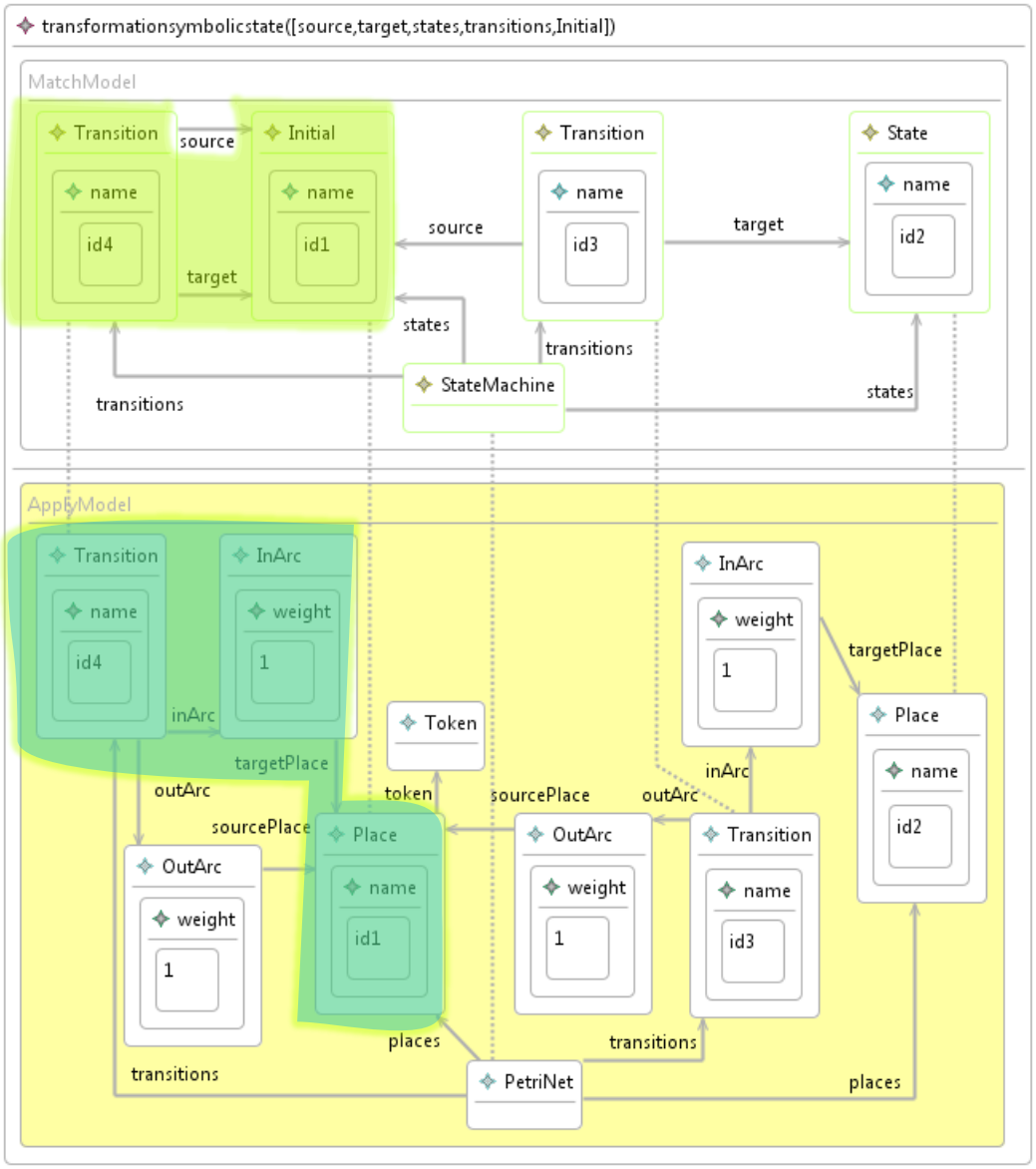


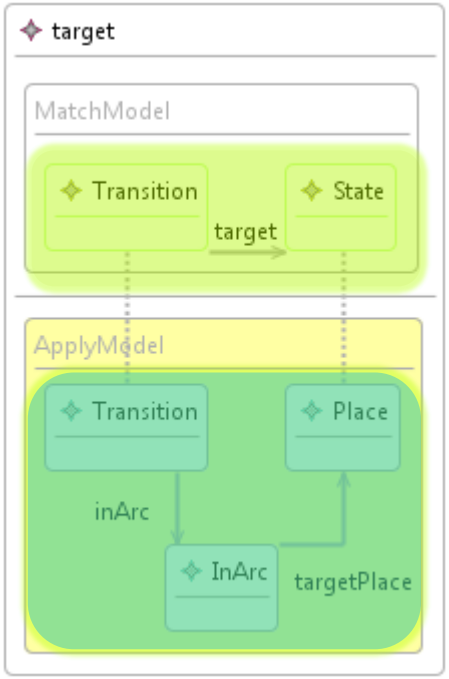
Experiment: Translation: Analysis I



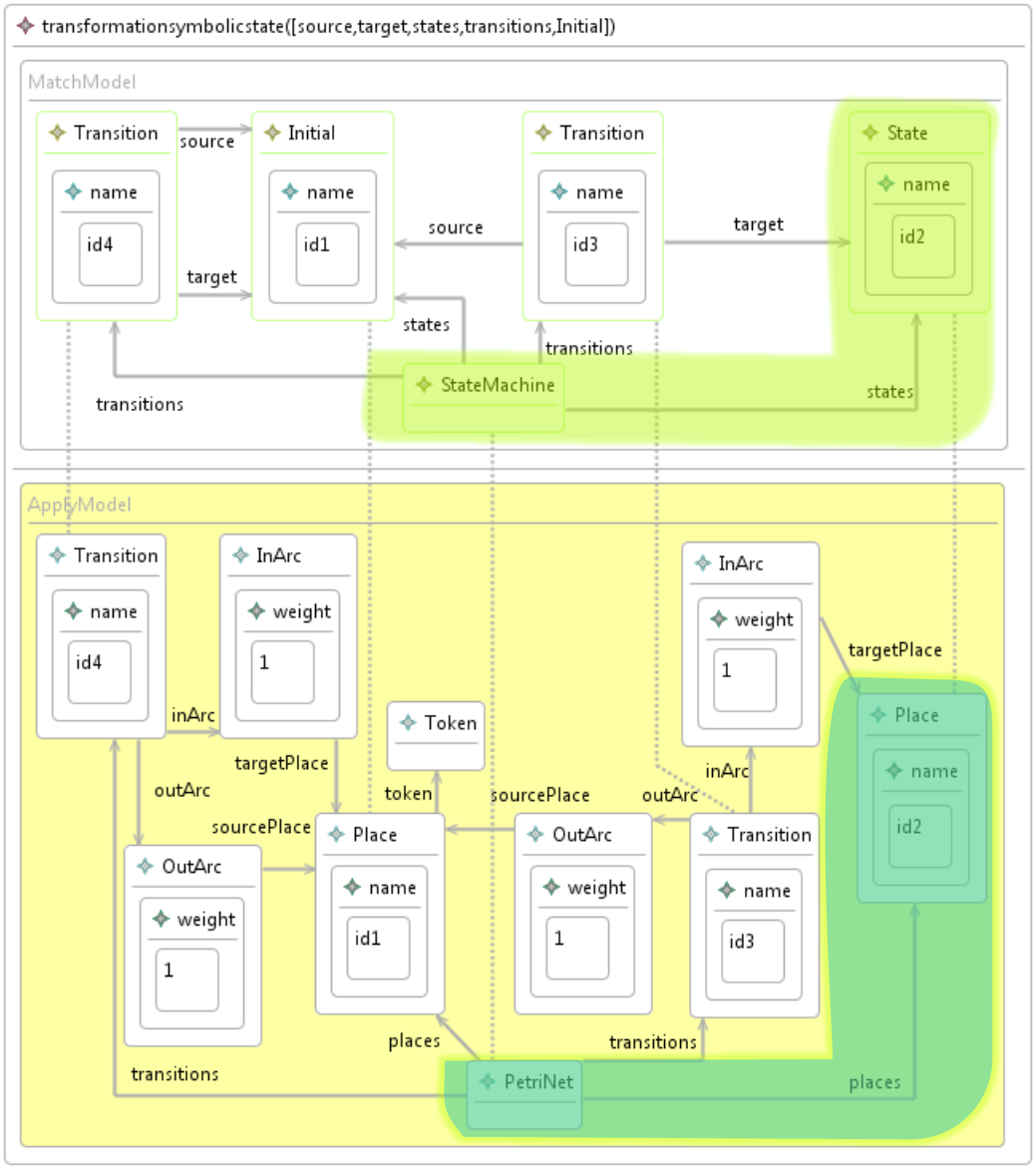


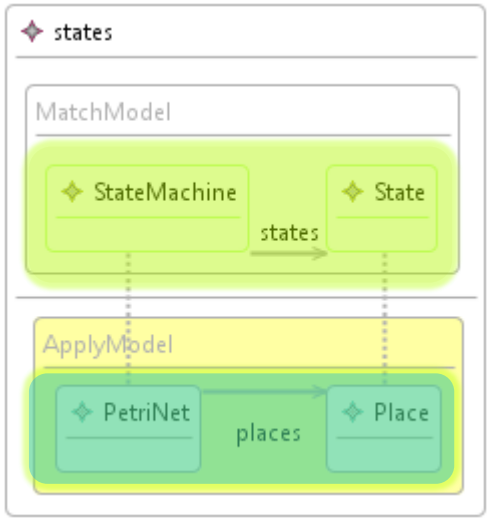
Experiment: Translation: Analysis I



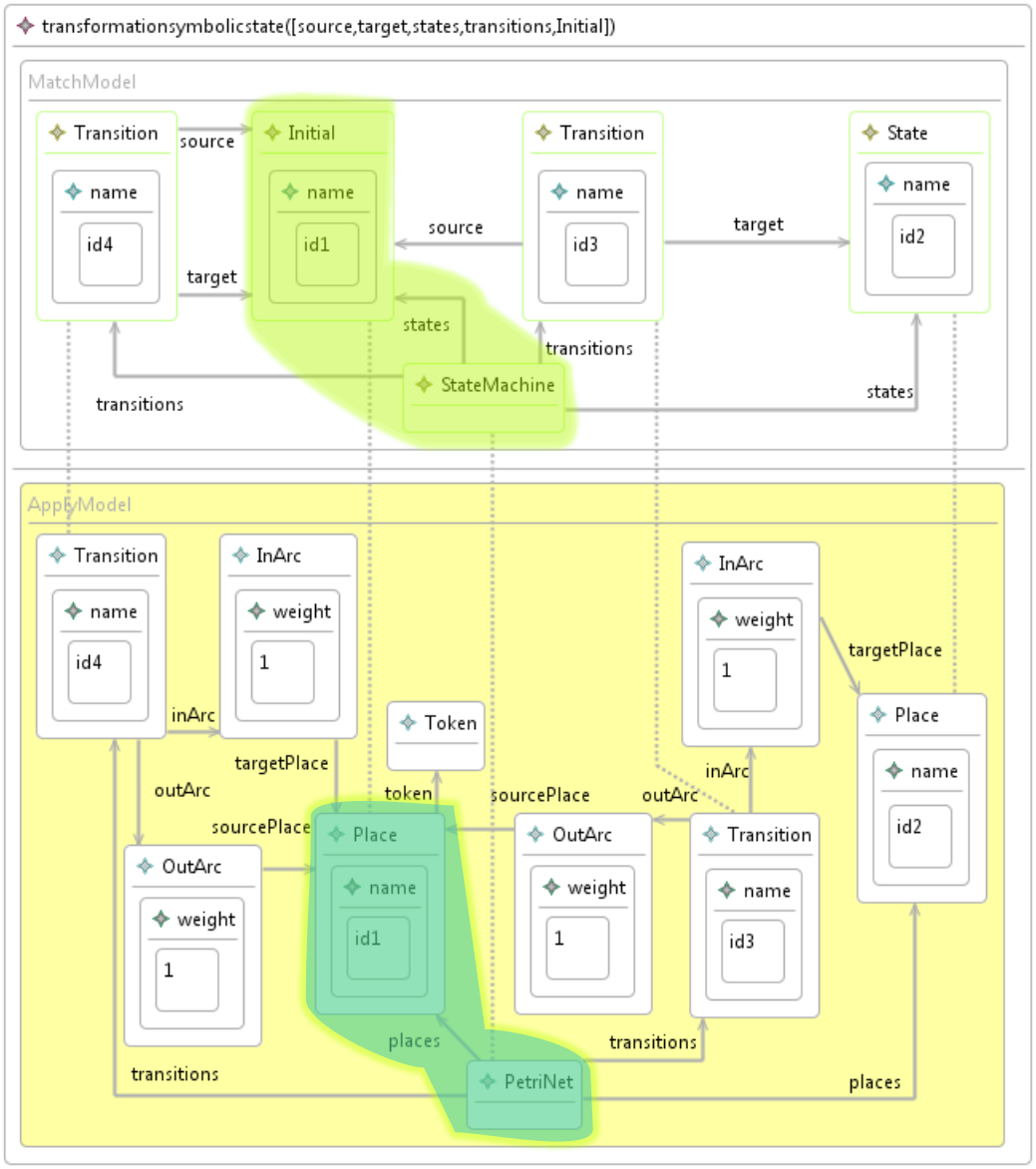


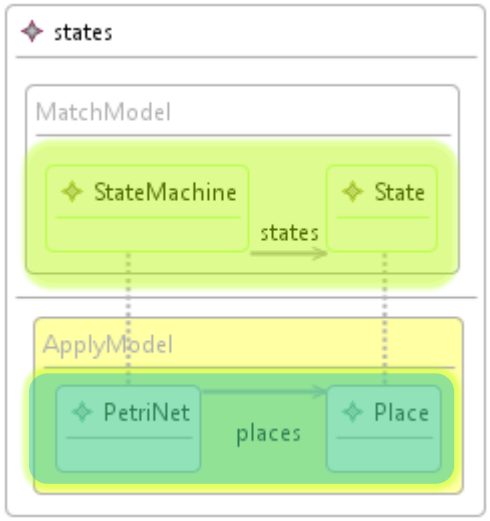
Experiment: Translation: Analysis I



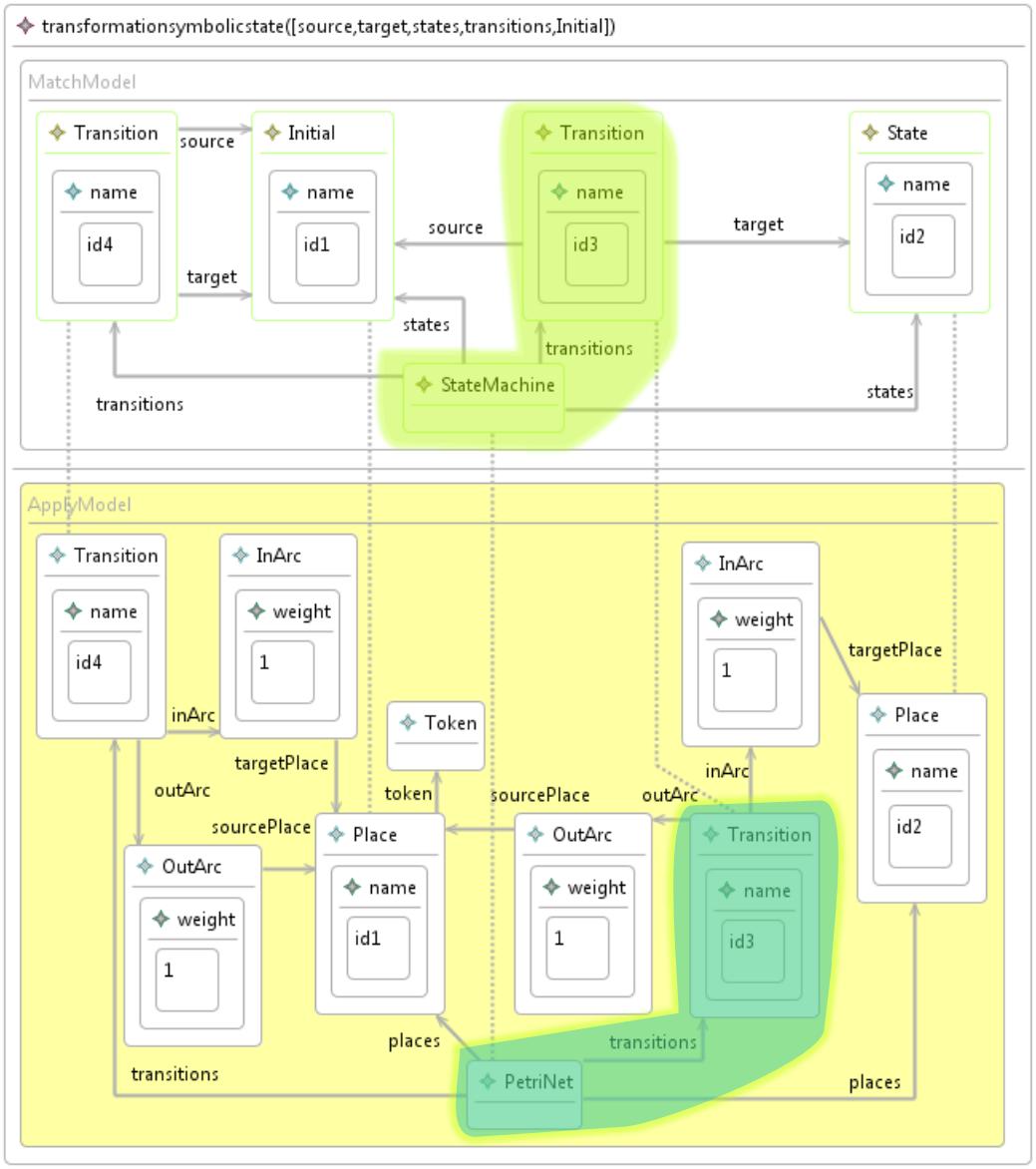


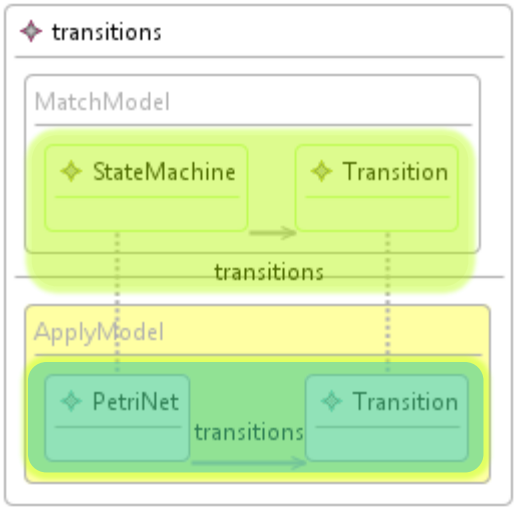
Experiment: Translation: Analysis I



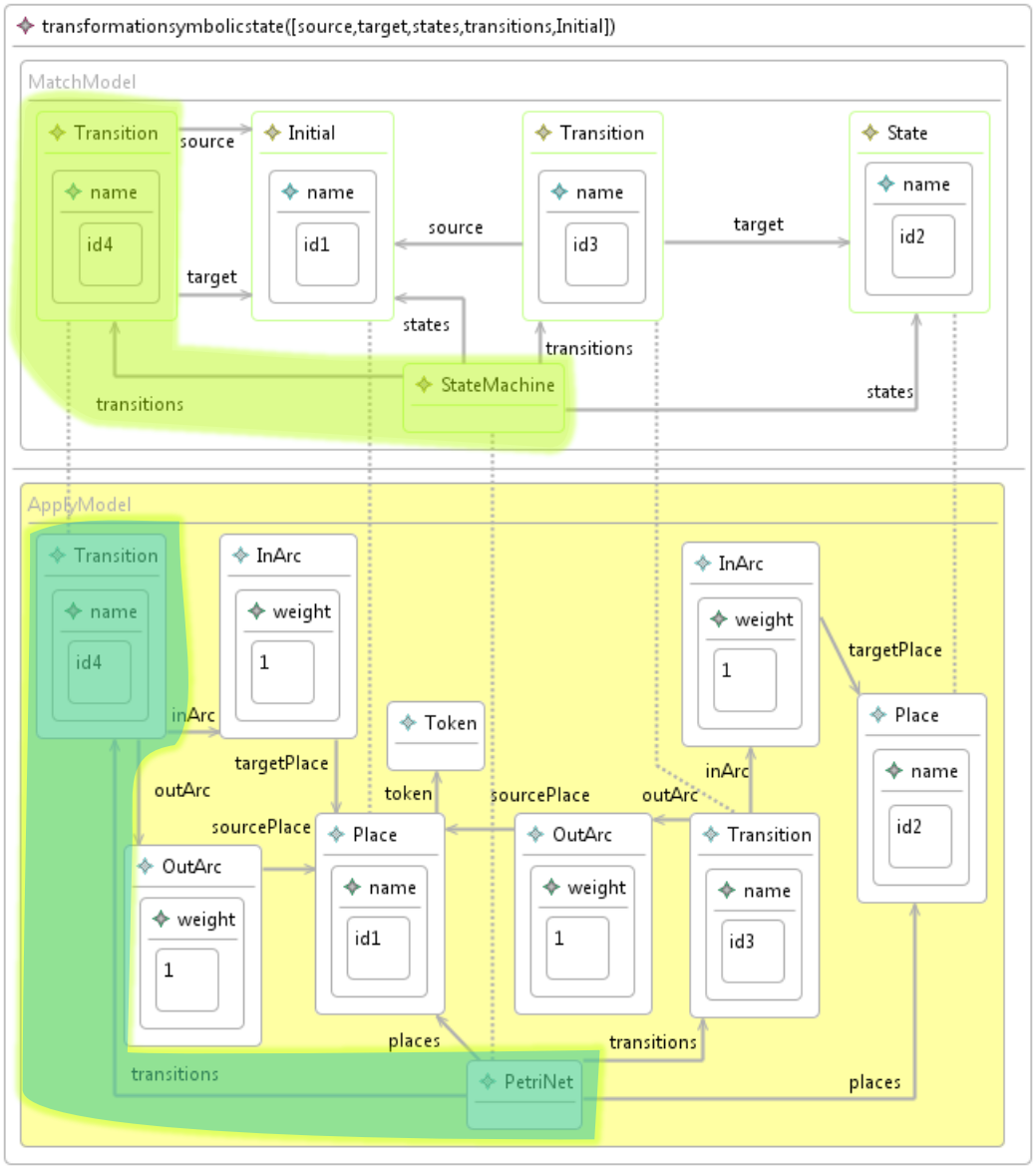


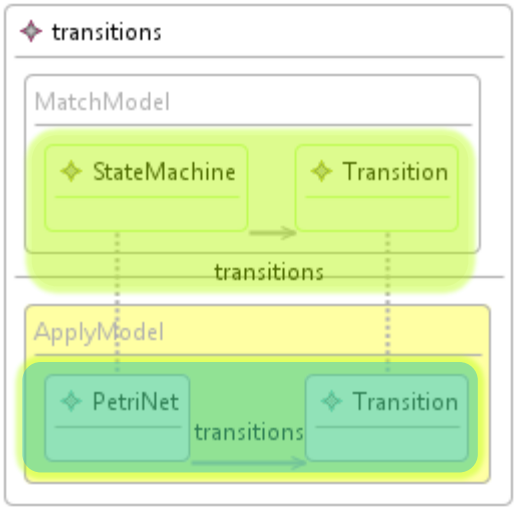
Experiment: Translation: Analysis I



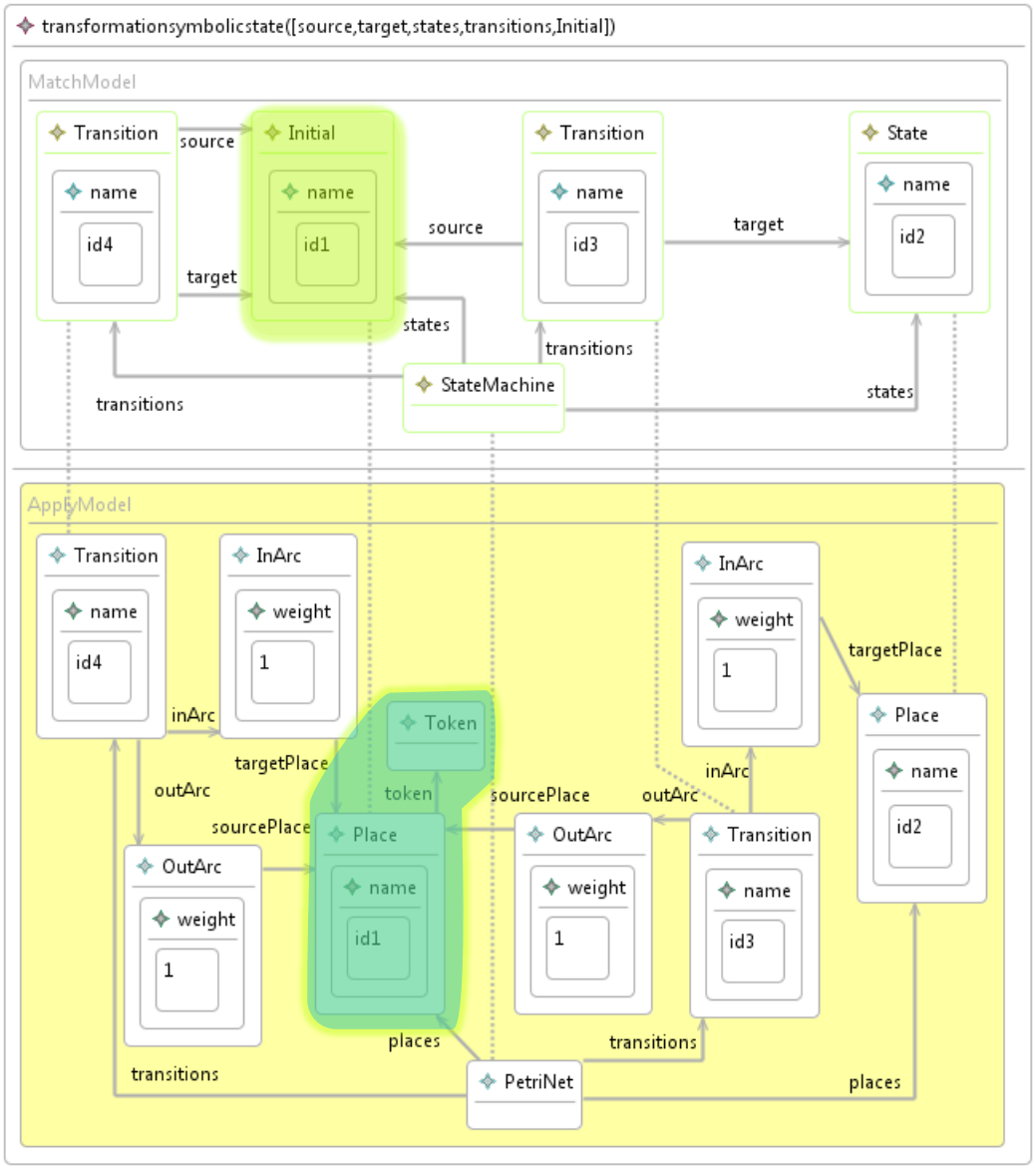


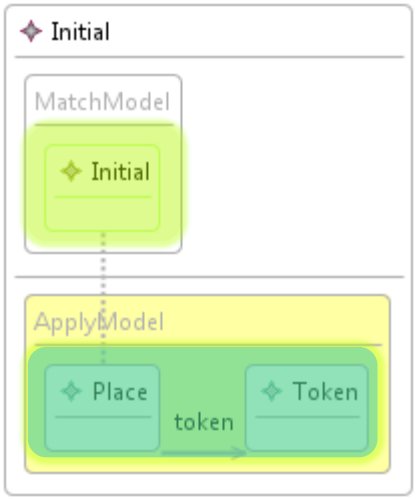
Experiment: Translation: Analysis I



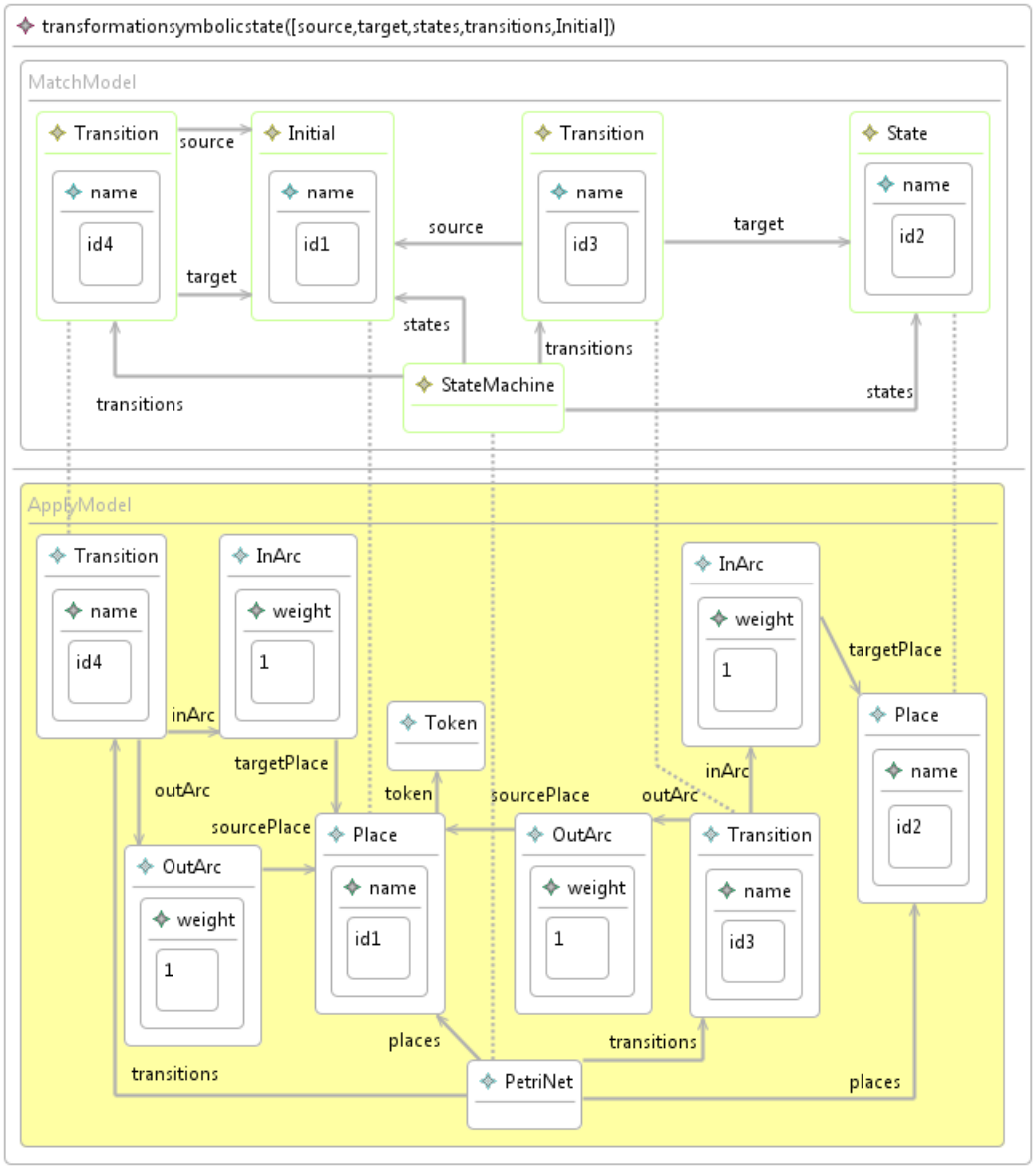


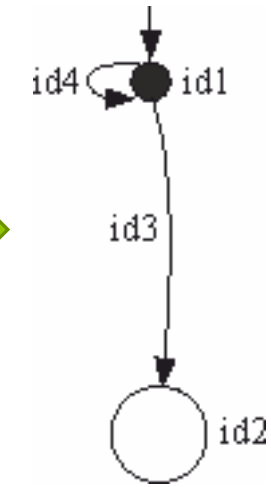
Experiment: Translation: Analysis I

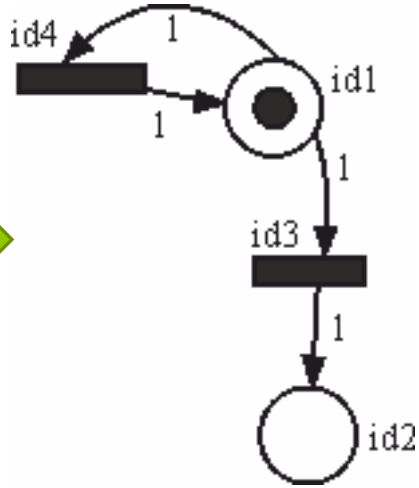




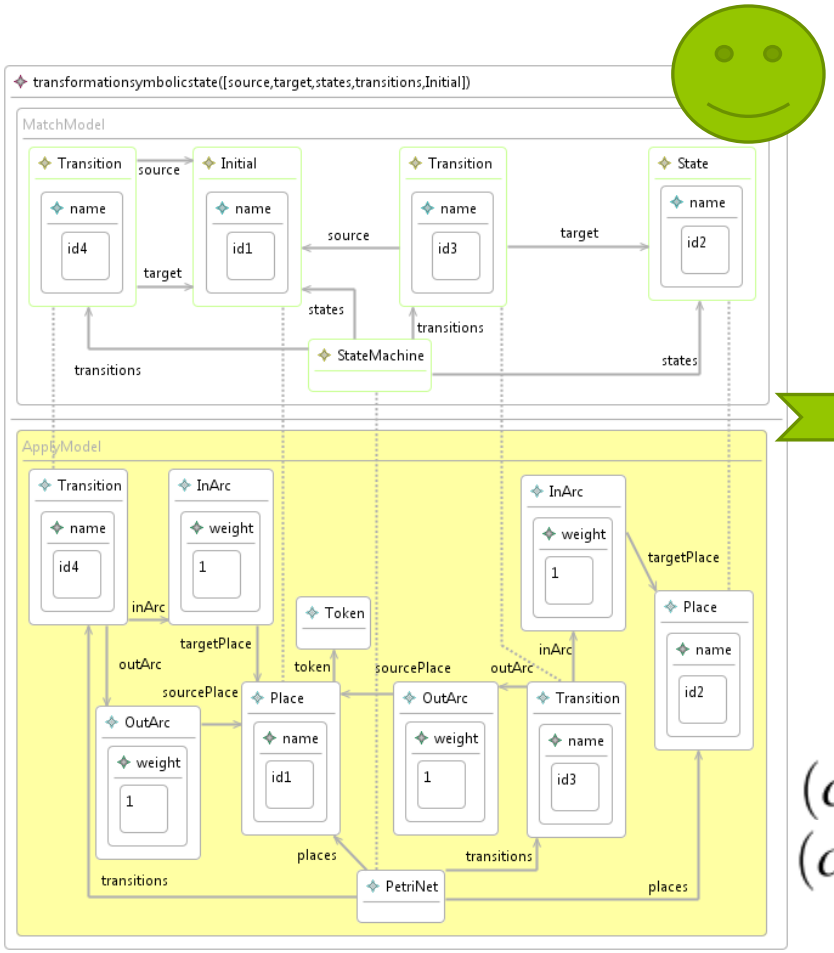
Experiment: Translation: Analysis I

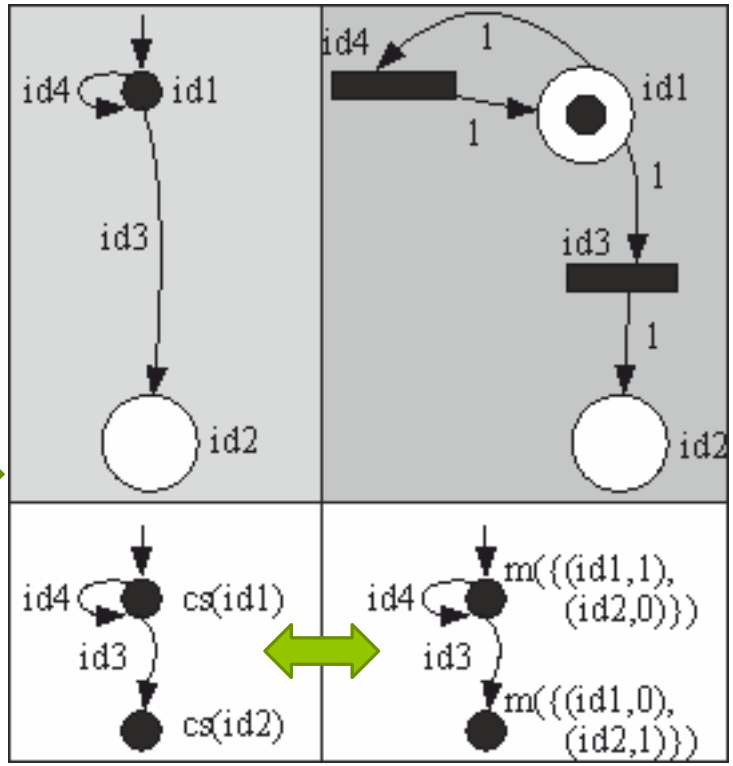






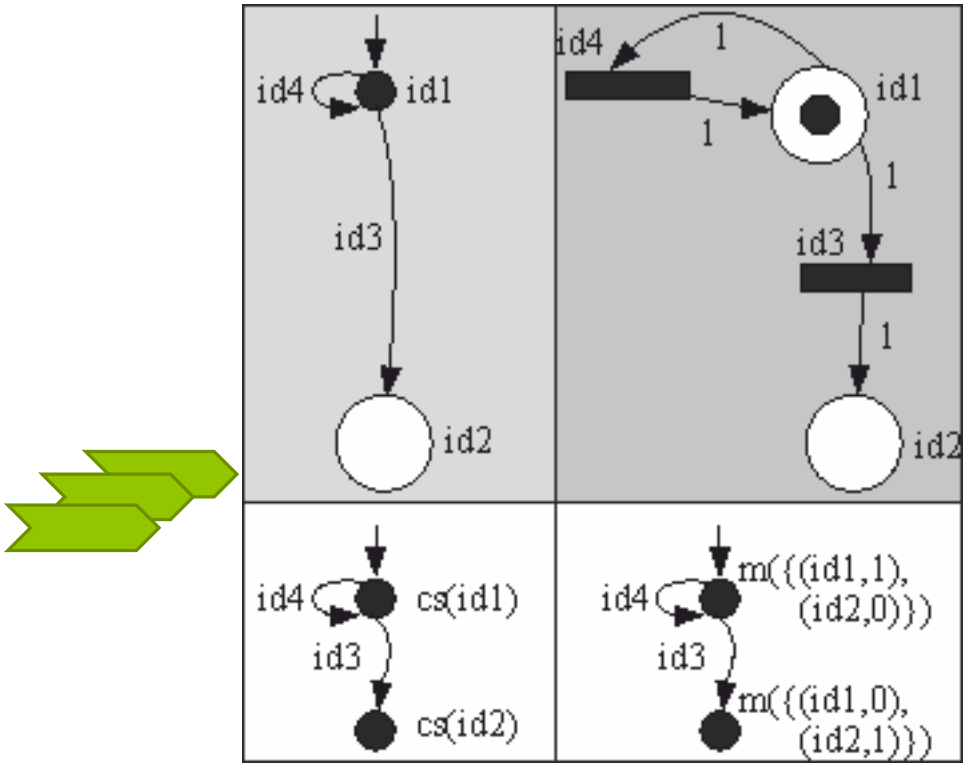
Experiment: Translation: Analysis II





$$(cs(id1)), m(\{(id1,1),(id2,0)\}) \in R$$
$$(cs(id2)), m(\{(id1,0),(id2,1)\}) \in R$$

Experiment: Translation: Analysis II



$$(cs(id1)), m(\{(id1,1),(id2,0)\}) \in R$$
$$(cs(id2)), m(\{(id1,0),(id2,1)\}) \in R$$
$$\ldots$$
$$(cs(State), m(\{(Trans(State),1)\}$$
$$\cup\{(Place',0) \mid Place'! = Trans(State)\}) \in R$$

Conclusions

- A technique to validate translations based on the involved SOS language semantics
- Explored the computation boundaries of the approach (in theory)
  - Finiteness of the translation's symbolic states
  - Finiteness of the resulting transition systems

Future Work

- Application of this approach to more interesting translations
  - Component Languages as target languages
  - DSMLs that can have infinite-sized transition systems

Questions