Winter Simulation Conference (WSC'04)
Washington, DC. December 5-8, 2004.

Computer Automated Multi-Paradigm Modelling for Analysis and Design of Traffic Networks

"model everything"

Hans Vangheluwe

School of Computer Science, McGill University, Montréal, Canada

Juan de Lara

E.T.S. de Informática, Universidad Autonóma de Madrid, Madrid, Spain

Overview

1. Computer Automated Multi-Paradigm Modelling (CAMPaM)
   $\Rightarrow$ Domain Specific (Visual) Modelling – DS(V)M
   - What/Why of DS(V)M (and DS(V)Ls) ?
2. Building DS(V)M Tools Effectively
   (a) Specifying textual/visual *syntax* of DS(V)Ls: *meta-modelling*
   (b) Specifying DS(V)L *semantics*: *transformations*
   (c) Modelling (and executing) *transformations*: *graph rewriting*
3. **Traffic**, a domain specific modelling formalism
   - Modelling a **Traffic**-Specific Modelling Tool
   - Various Transformations
4. Conclusions and Future Work

Computer Automated Multi-Paradigm Modelling (CAMPaM)

1. Different *levels of abstraction*
2. Mixing *different modelling formalisms* (coupling, transformation)
3. Modelling classes of models (formalisms) by *meta-modelling*
4. Modelling *model transformations* explicitly

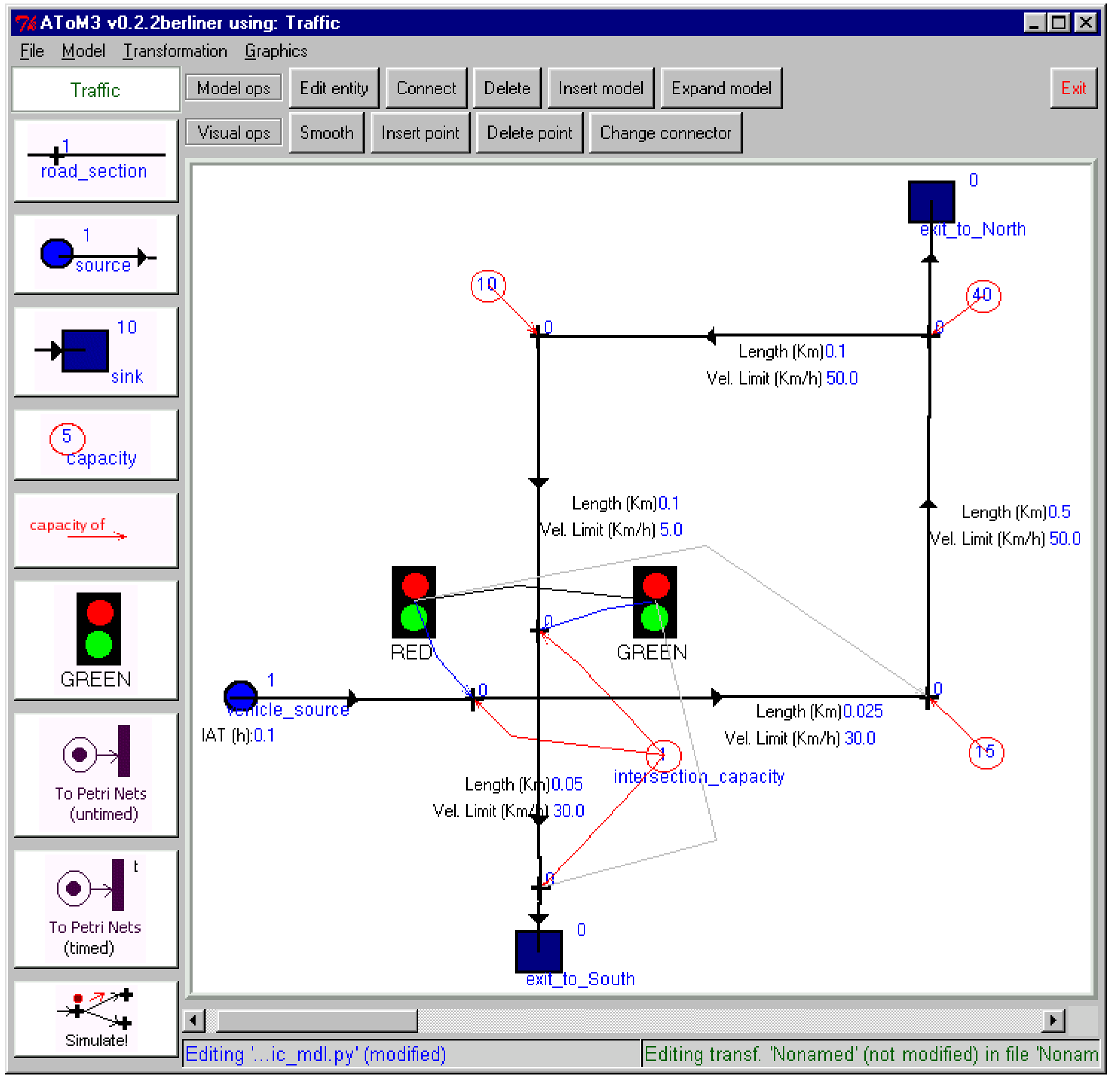
Pieter J. Mosterman and Hans Vangheluwe.

Computer Automated Multi-Paradigm Modeling: An Introduction.

Simulation: Transactions of the Society for Modeling and Simulation International, 80(9), September 2004. Special Issue: Grand Challenges for Modeling and Simulation.

A Simple Example to Demonstrate Concepts: Model/Analyze/Simulate Traffic Networks

Approach: Domain Specific (Visual) Modelling



Why DS(V)M ?
(as opposed to General Purpose modelling)

- **match the user's mental model** of the problem domain
- **maximally constrain** user (to the problem at hand)
  $\Rightarrow$ easier to learn
  $\Rightarrow$ avoid errors
- **separate** domain-expert's work
  from analysis/transformation expert's work
- re-use **transformation** knowledge
  (*e.g.*, in variations of a domain specific formalism)

Building DS(V)M Tools Effectively . . .

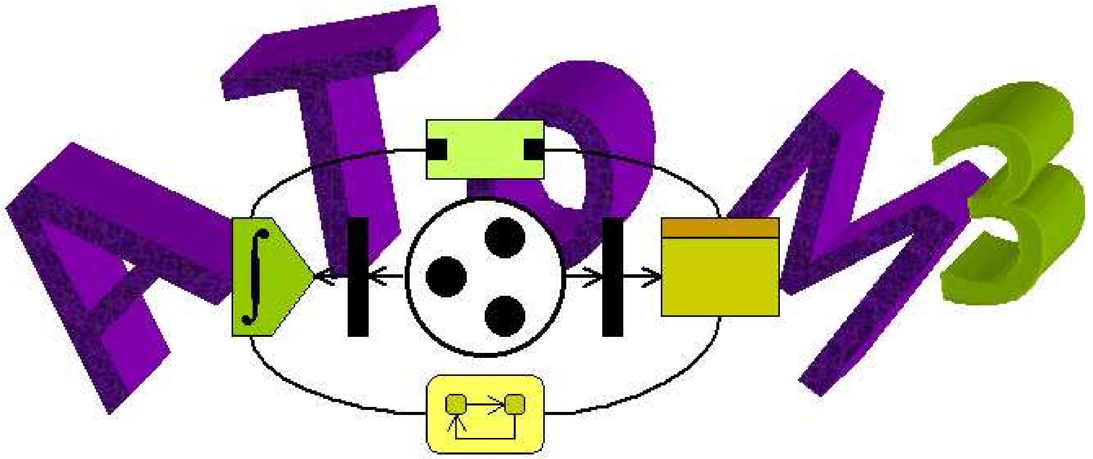
- **development cost** of DS(V)M Tools may be prohibitive !
- we want to effectively (rapidly, correctly, re-usably, . . .) **specify** and **generate/execute**:
  - Domain Specific (Visual) **Languages** (DS(V)Ls)
  - (reactive) **behaviour** of DS(V)M environments/tools
  - **model transformations** (for analysis, optimization, simulation . . .)

$\Rightarrow$ **model everything**

How to Build DS(V)M Tools Effectively ?

1. Specify textual/visual **syntax** of DS(V)Ls:
   meta-modelling
2. Specify DS(V)L **semantics**:
   transformation
3. Model (and analyze and execute) **transformations**:
   graph rewriting

A Tool for Multi-formalism and Meta-Modelling

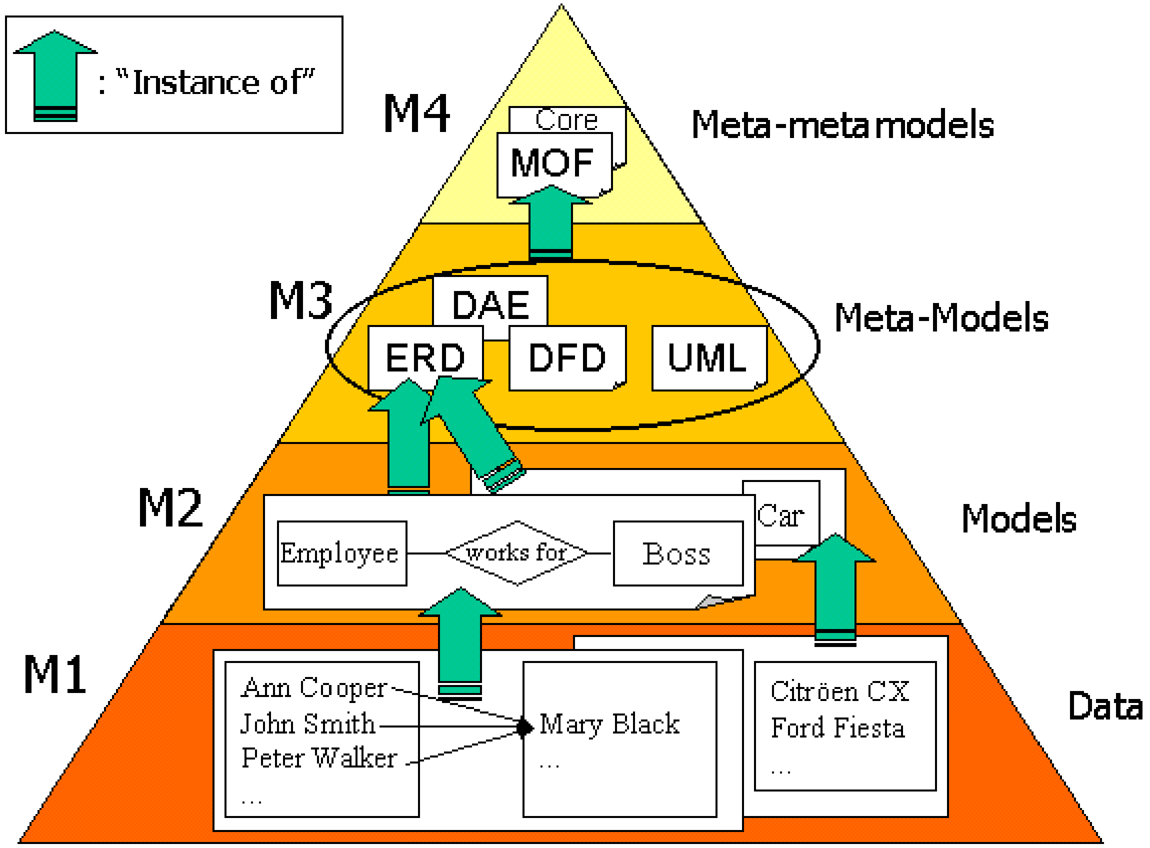


atom3.cs.mcgill.ca

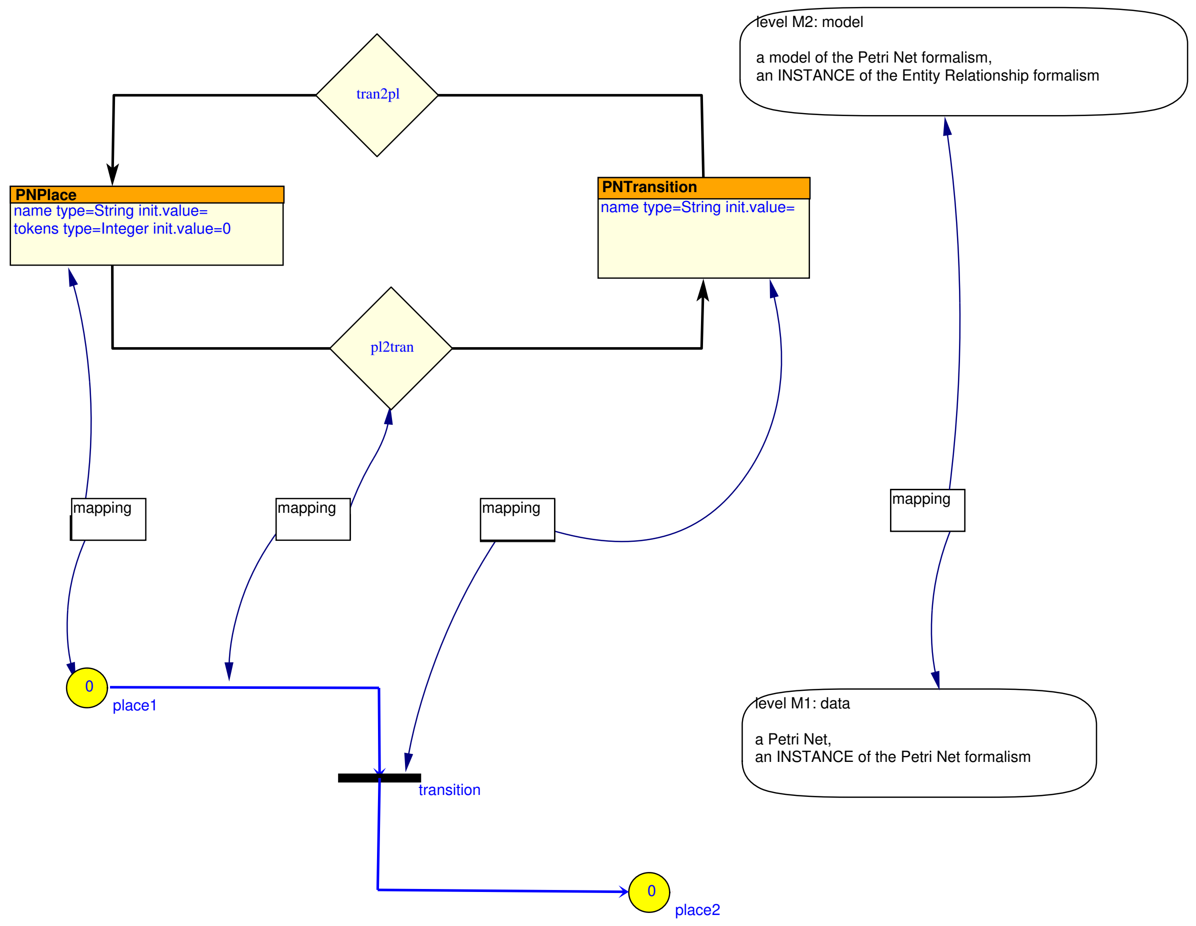
Specifying textual/visual *syntax* of DS(V)Ls

- **abstract** syntax:
  - syntax grammar (text grammar, AToM^3 Graph Grammar) or
  - **meta-model** ($\sim$ type graph)
- **concrete** syntax:
  - textual (lexical specification) or
  - **visual** (AToM^3 "icons" $+$ connections)

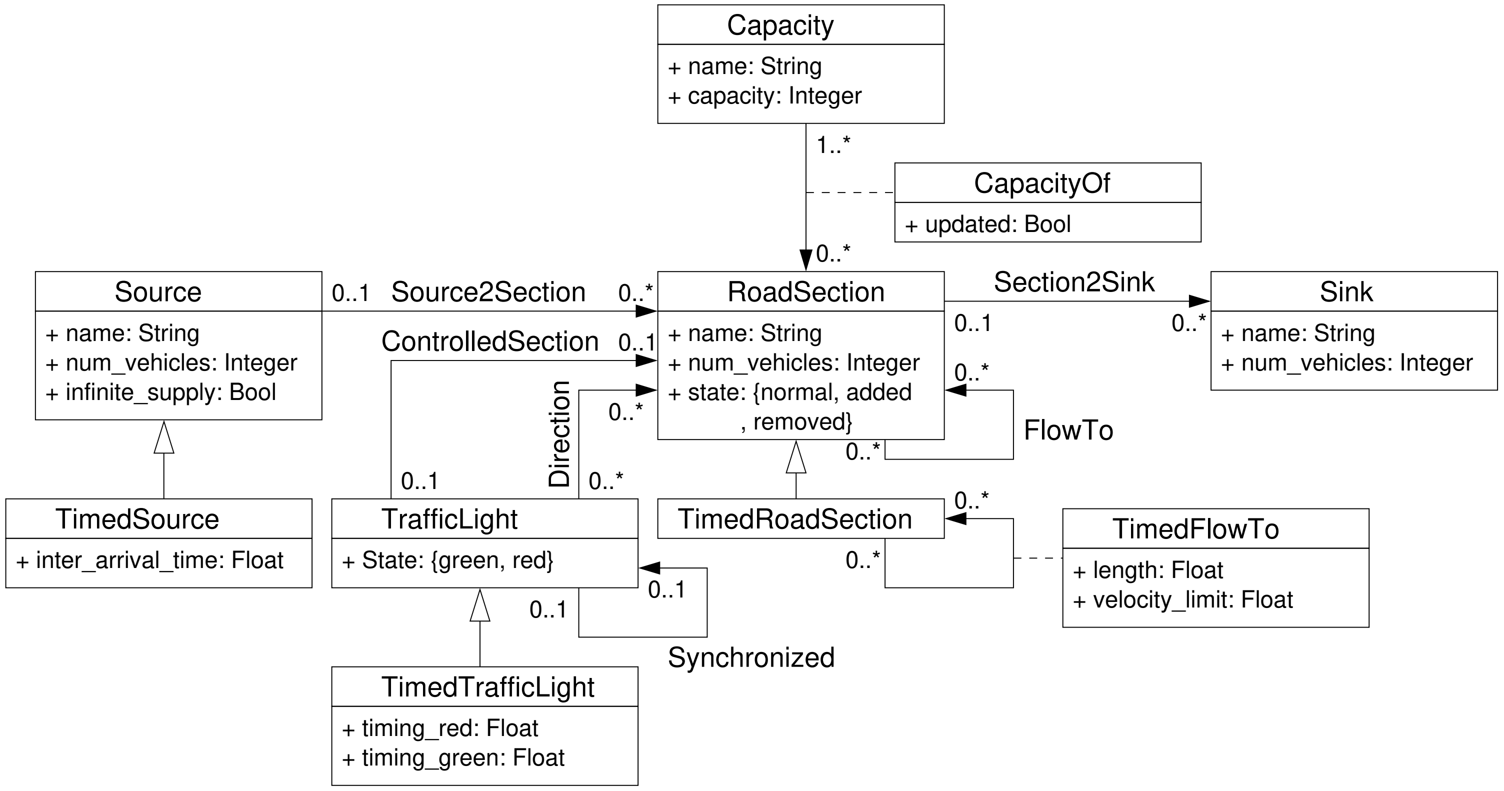
Meta-modelling (OMG-style)



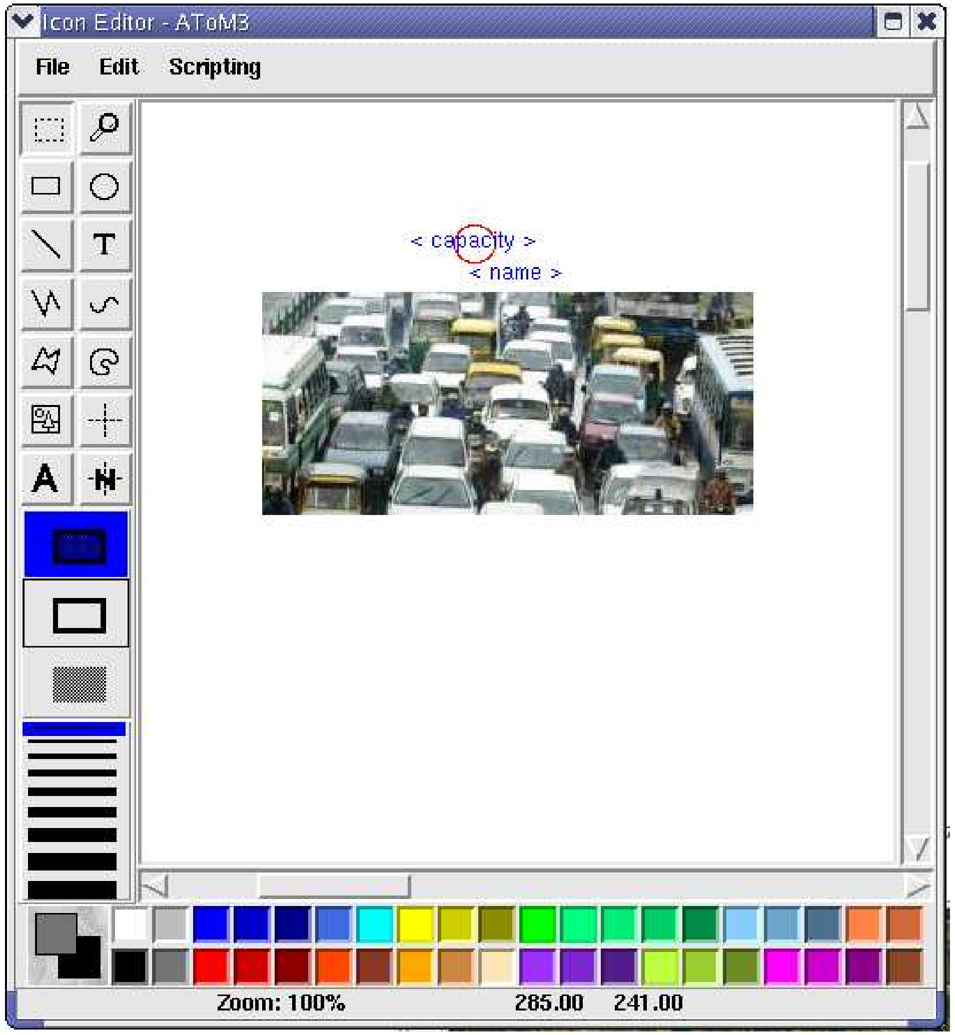
Meta-modelling: model-instance morphism


tran2pl
PNPlace
name type=String init.value=
tokens type=Integer init.value=0
PNTransition
name type=String init.value=
pl2tran
level M2: model
a model of the Petri Net formalism,
an INSTANCE of the Entity Relationship formalism
mapping
mapping
mapping
mapping
0
place1
transition
0
place2
level M1: data
a Petri Net,
an INSTANCE of the Petri Net formalism

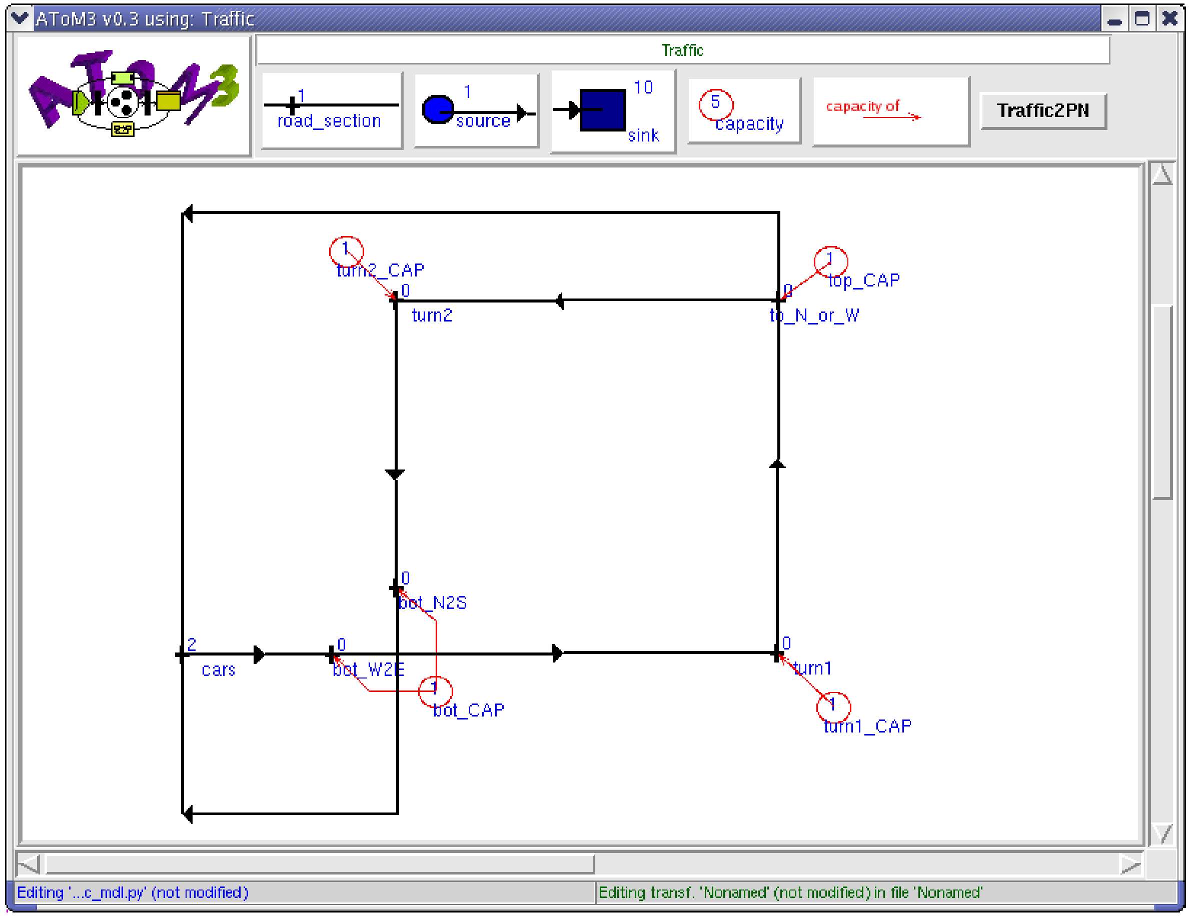
Un-timed and timed Traffic Formalism meta-model (a model in the UML Class Diagram Formalism)



Traffic Concrete Syntax (the Capacity Entity)



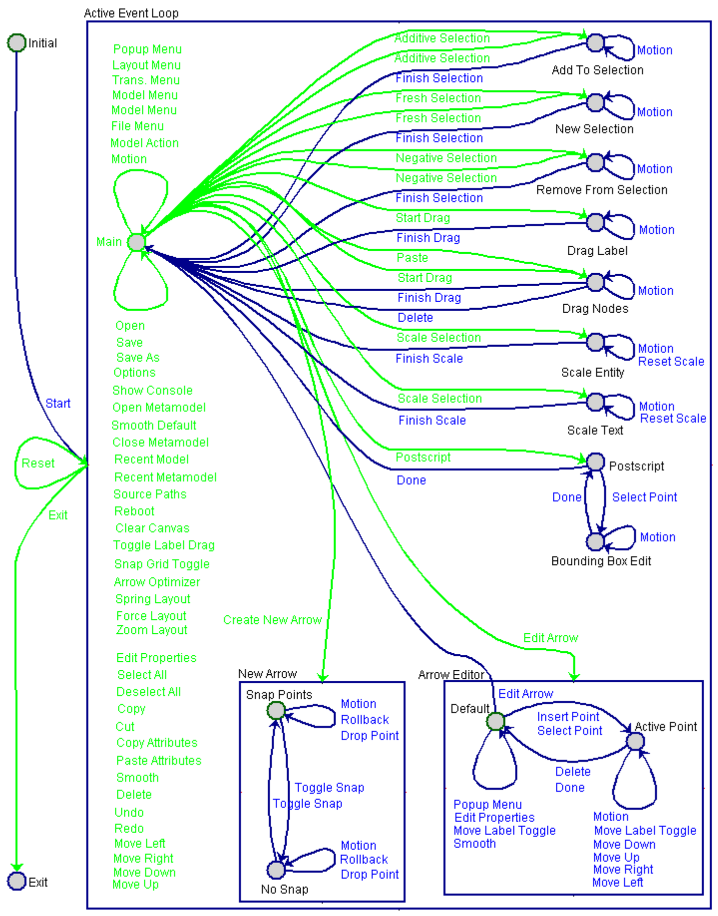
The generated Traffic visual modelling environment

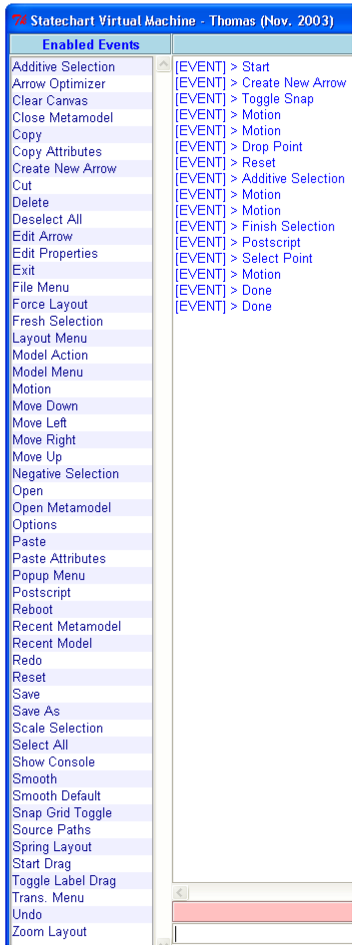


Editing '...c_mdl.py' (not modified)

Editing transf. 'Nonamed' (not modified) in file 'Nonamed'

Caveat: Statechart model of the GUI's Reactive Behaviour



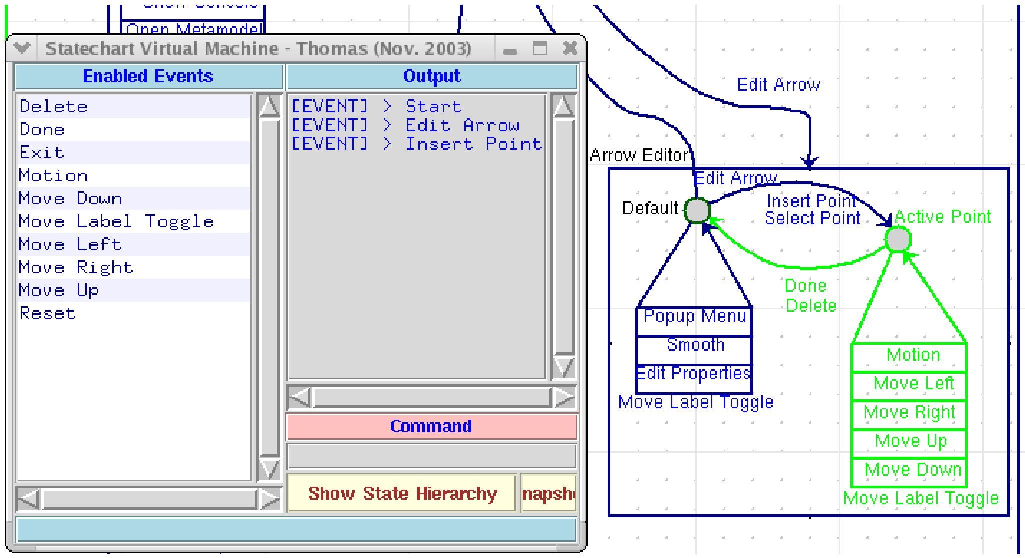

Statechart Virtual Machine - Thomas (Nov. 2003)

Enabled Events

Additive Selection
Arrow Optimizer
Clear Canvas
Close Metamodel
Copy
Copy Attributes
Create New Arrow
Cut
Delete
Deselect All
Edit Arrow
Edit Properties
Exit
File Menu
Force Layout
Fresh Selection
Layout Menu
Model Action
Model Menu
Motion
Move Down
Move Left
Move Right
Move Up
Negative Selection
Open
Open Metamodel
Options
Paste
Paste Attributes
Popup Menu
Postscript
Reboot
Recent Metamodel
Recent Model
Redo
Reset
Save
Save As
Scale Selection
Select All
Show Console
Smooth
Smooth Default
Snap Grid Toggle
Source Paths
Spring Layout
Start Drag
Toggle Label Drag
Trans. Menu
Undo
Zoom Layout

```
[EVENT] > Start
[EVENT] > Create New Arrow
[EVENT] > Toggle Snap
[EVENT] > Motion
[EVENT] > Motion
[EVENT] > Drop Point
[EVENT] > Reset
[EVENT] > Additive Selection
[EVENT] > Motion
[EVENT] > Motion
[EVENT] > Finish Selection
[EVENT] > Postscript
[EVENT] > Select Point
[EVENT] > Motion
[EVENT] > Done
[EVENT] > Done
```

The GUI's reactive behaviour in action



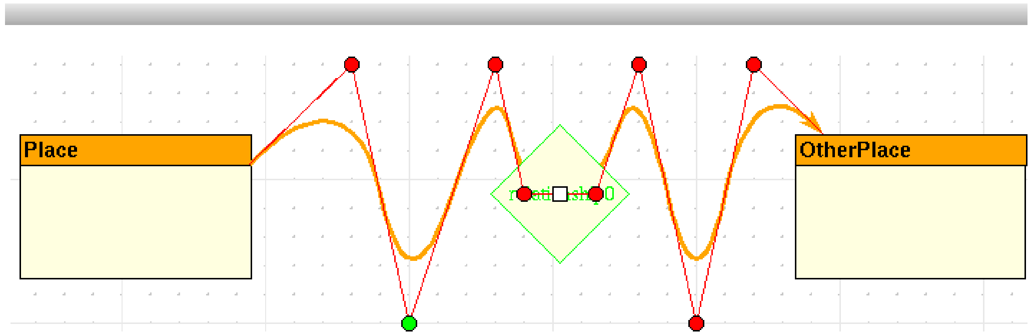
Statechart Virtual Machine - Thomas (Nov. 2003)

Enabled Events

```
Delete
Done
Exit
Motion
Move Down
Move Label Toggle
Move Left
Move Right
Move Up
Reset
```

Output

```
[EVENT] > Start
[EVENT] > Edit Arrow
[EVENT] > Insert Point
```

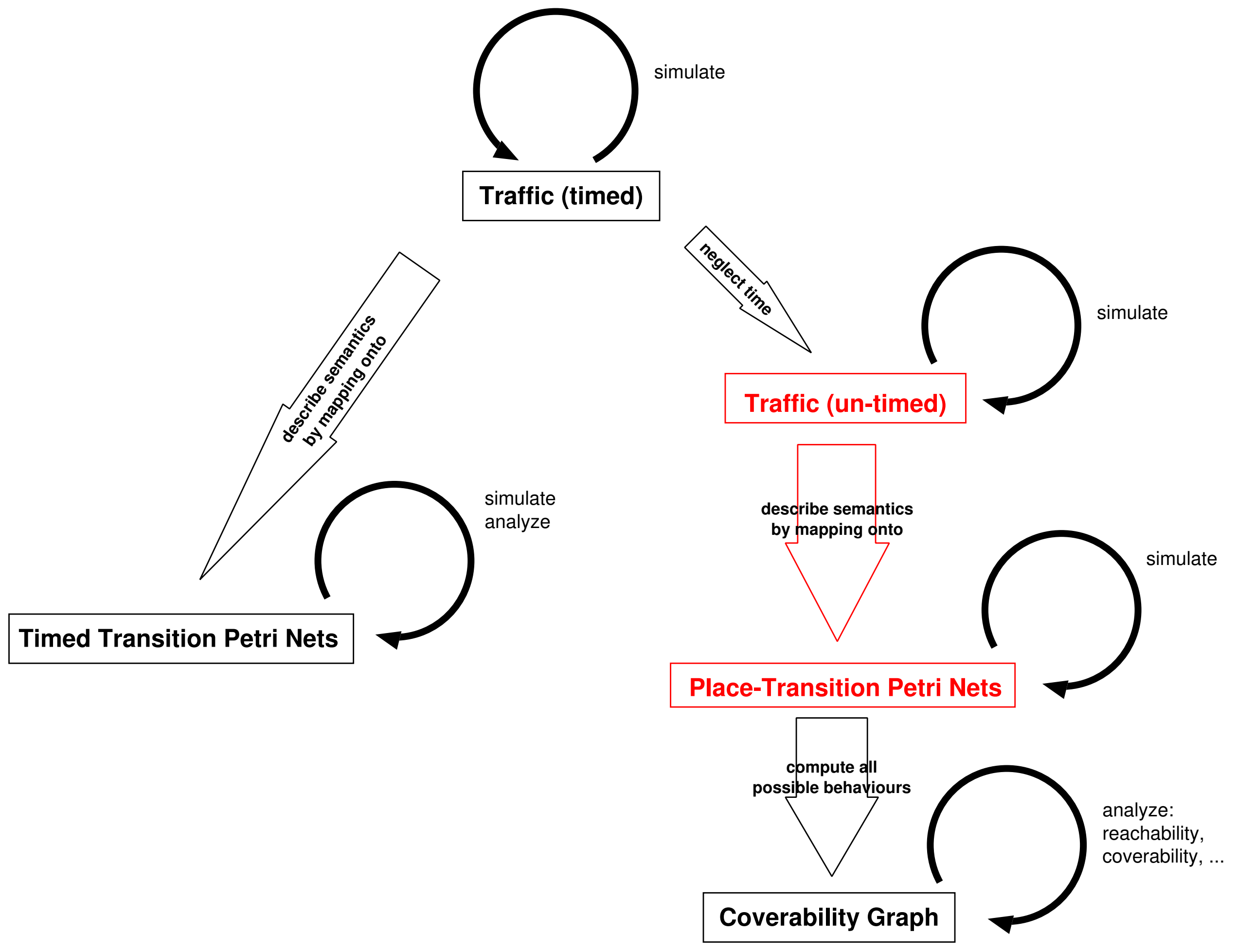


current work: what is the *optimal* formalism to specify GUI reactive behaviour ?

Modelling Traffic's Semantics

- choices: timed, un-timed, . . . (level of abstraction)
- **denotational**: map onto known formalism (TTPN, PN)
  . . . good for analysis purposes
- **operational**: procedure to execute/simulate model
  . . . may act as a reference implementation
- note: need to *prove* consistency between denotational and operational semantics if both are given !

Traffic, the Big Picture


simulate
Traffic (timed)
neglect time
describe semantics by mapping onto
simulate
Traffic (un-timed)
simulate analyze
Timed Transition Petri Nets
describe semantics by mapping onto
simulate
Place-Transition Petri Nets
compute all possible behaviours
analyze: reachability, coverability, ...
Coverability Graph

Traffic's (un-timed) semantics in terms of Petri Nets

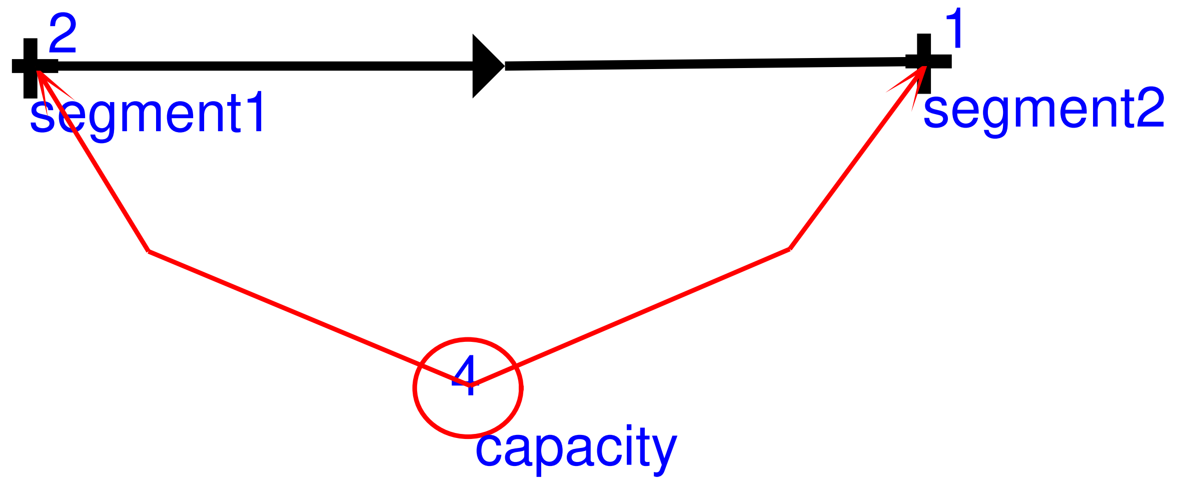
- need a meta-model of Traffic (shown before)
- need a meta-model of Petri Nets (shown before)
- need a model of the mapping: Traffic $\Rightarrow$ Petri Nets

Graph Transformation for Model Transformations

Ehrig, H., G. Engels, H.-J. Kreowski, and G. Rozenberg.
Handbook of graph grammars and computing by graph transformation.
1999. World Scientific.

Tools:
AGG, PROGRES, GME, AToM^3, Fujaba, . . .

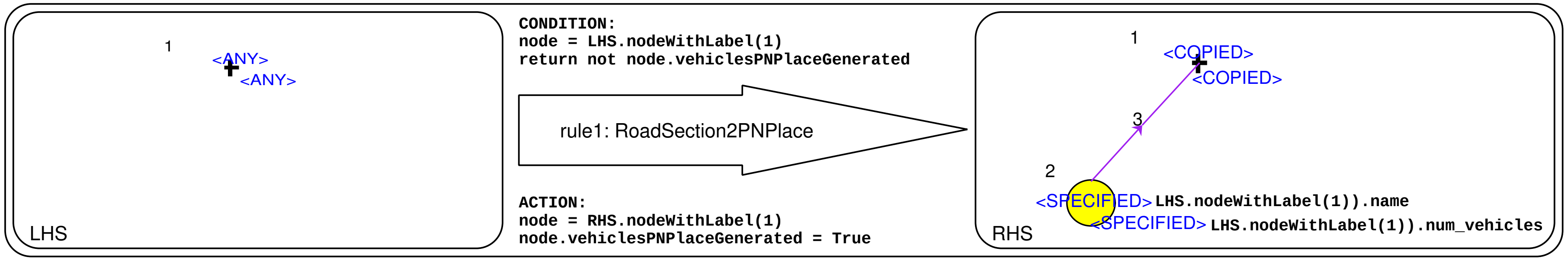
A very simple Traffic model



Traffic to Petri Net Graph Transformation rules

```
INITIAL ACTION:
for node in graph.listNodes["RoadSection"]:
 node.vehiclesPNPlaceGenerated=False
```

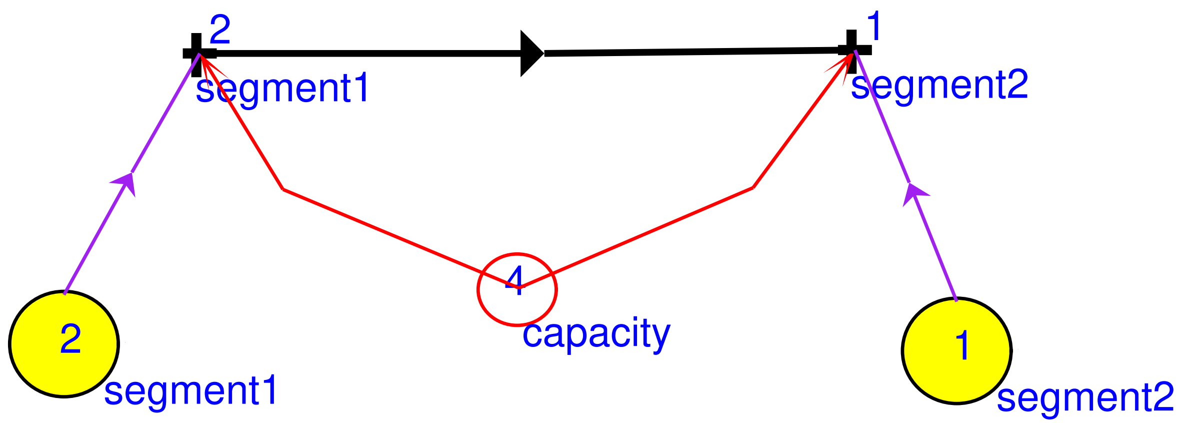
Traffic to Petri Net Graph Transformation rules


1
<ANY>
<ANY>
LHS
rule1: RoadSection2PNPlace
1
<COPIED>
<COPIED>
3
2
<SPECIFIED> LHS.nodeWithLabel(1)).name
<SPECIFIED> LHS.nodeWithLabel(1)).num_vehicles
RHS

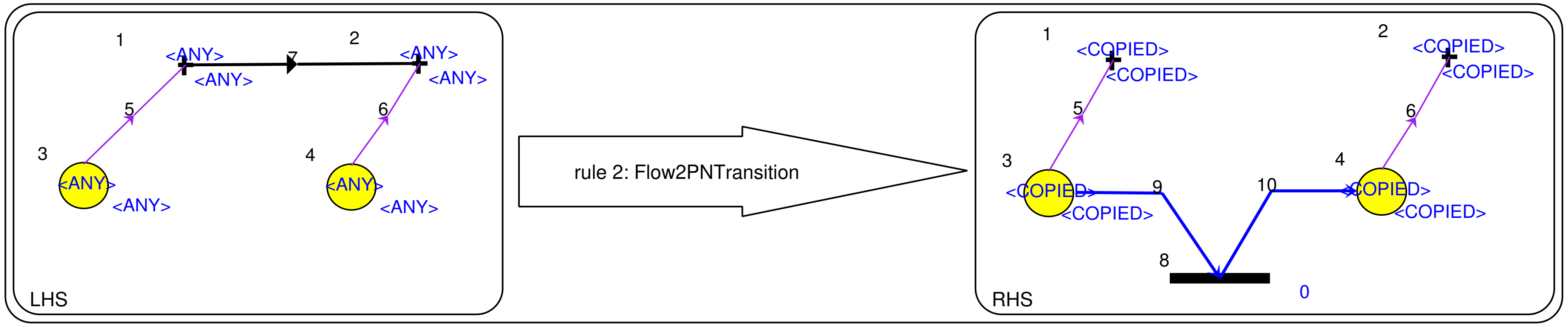
```
CONDITION:
node = LHS.nodeWithLabel(1)
return not node.vehiclesPNPlaceGenerated
```

```
ACTION:
node = RHS.nodeWithLabel(1)
node.vehiclesPNPlaceGenerated = True
```

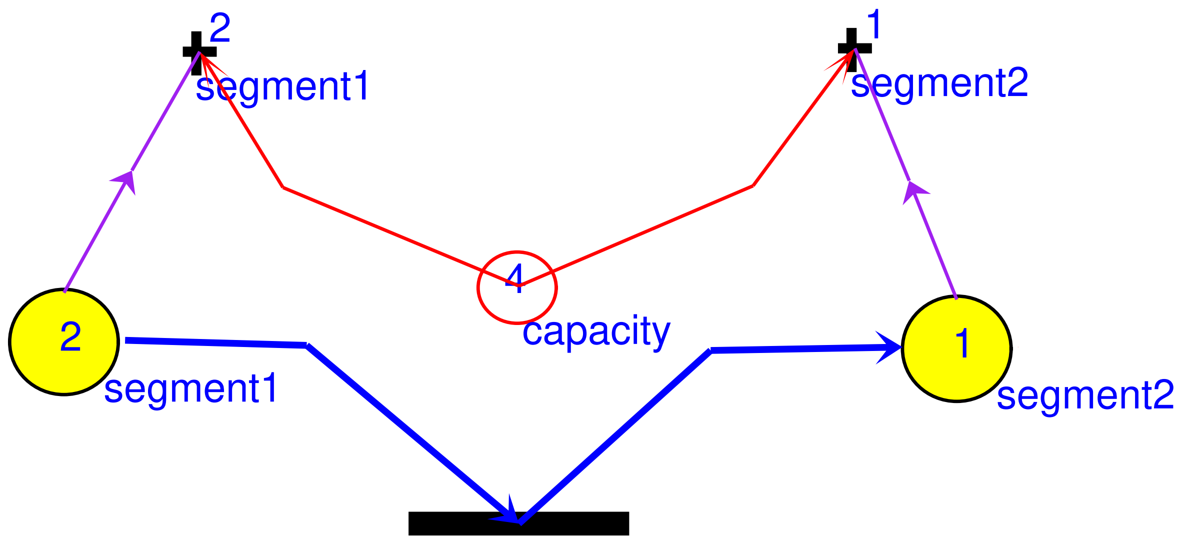
Road Sections converted to Petri Net Places



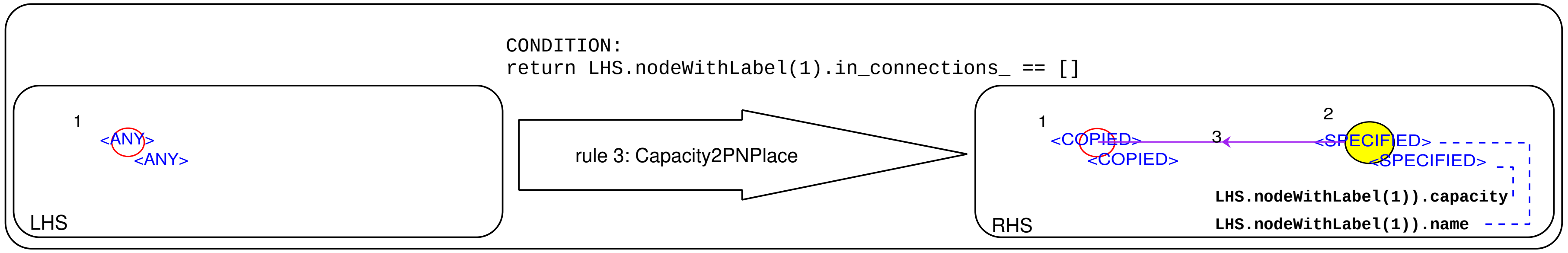
Traffic to Petri Net Graph Transformation rules



Traffic Flow to Petri Net Transitions

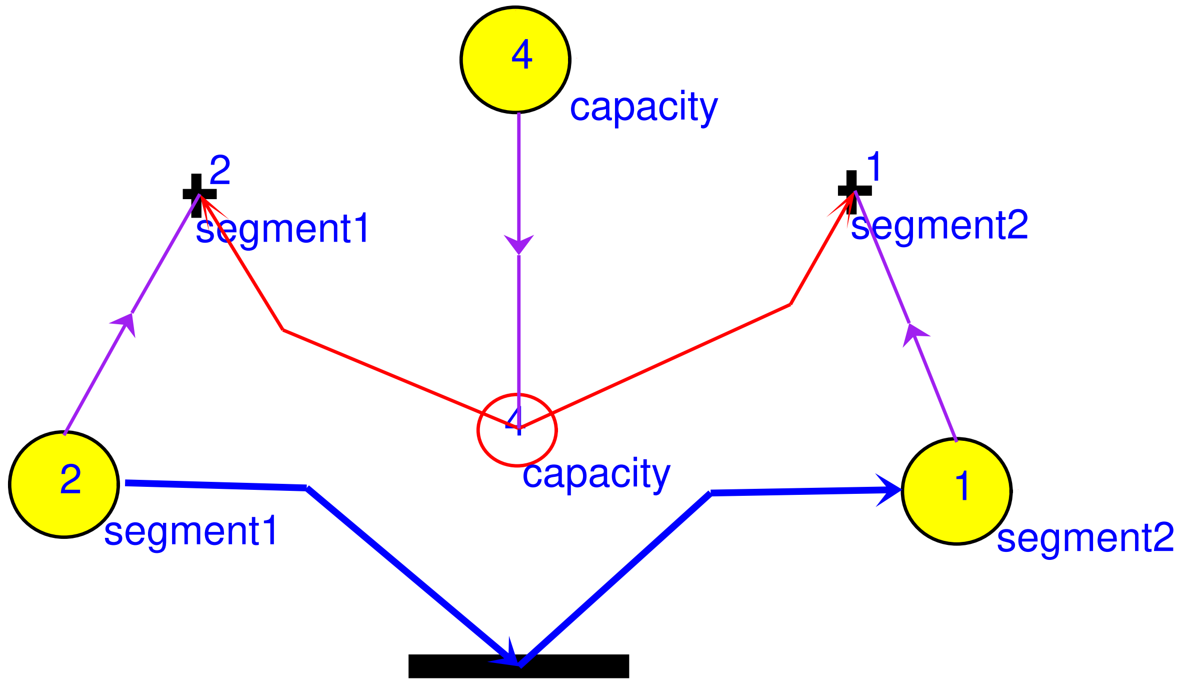


Traffic to Petri Net Graph Transformation rules

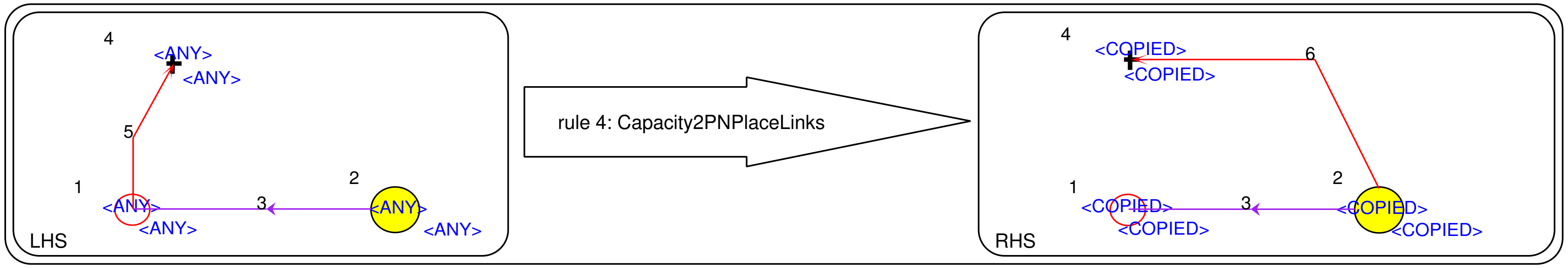


CONDITION:
return LHS.nodeWithLabel(1).in_connections_ == []
1
<ANY>
<ANY>
LHS
rule 3: Capacity2PNPlace
1
<COPIED>
<COPIED>
3
2
<SPECIFIED>
<SPECIFIED>
LHS.nodeWithLabel(1)).capacity
LHS.nodeWithLabel(1)).name
RHS

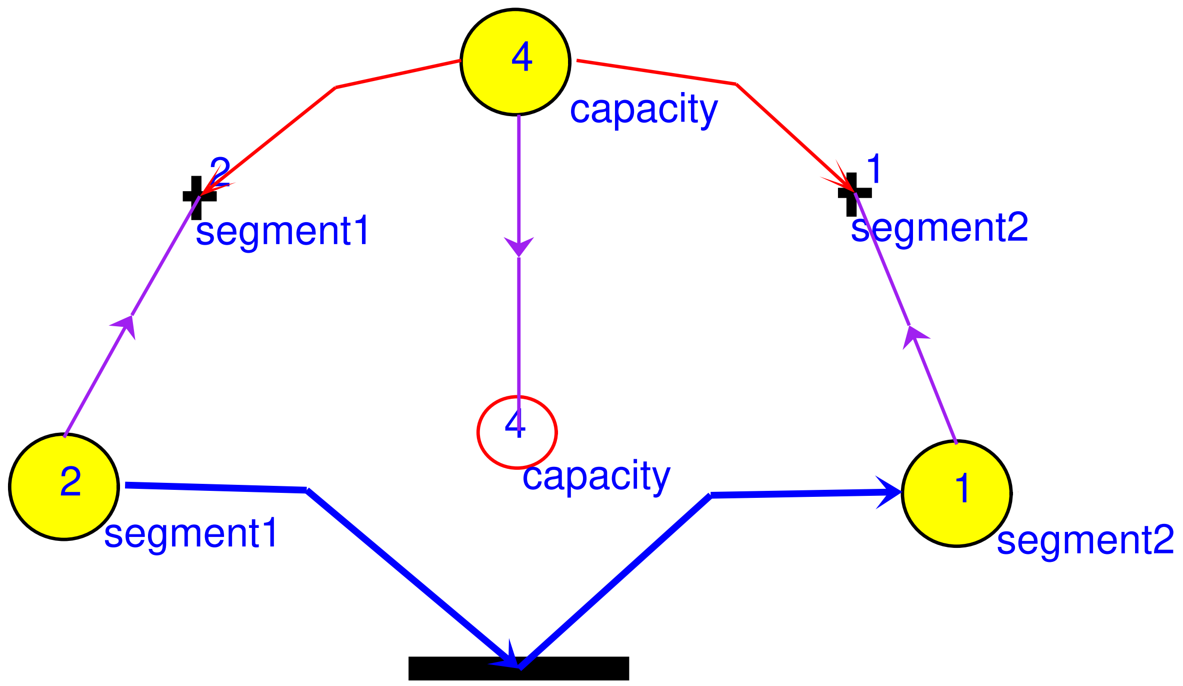
Traffic Capacity to Petri Net Place



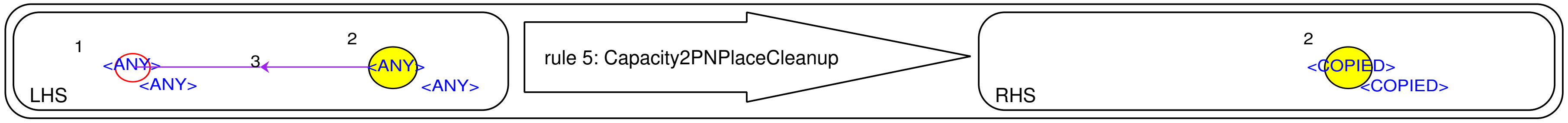
Traffic to Petri Net Graph Transformation rules



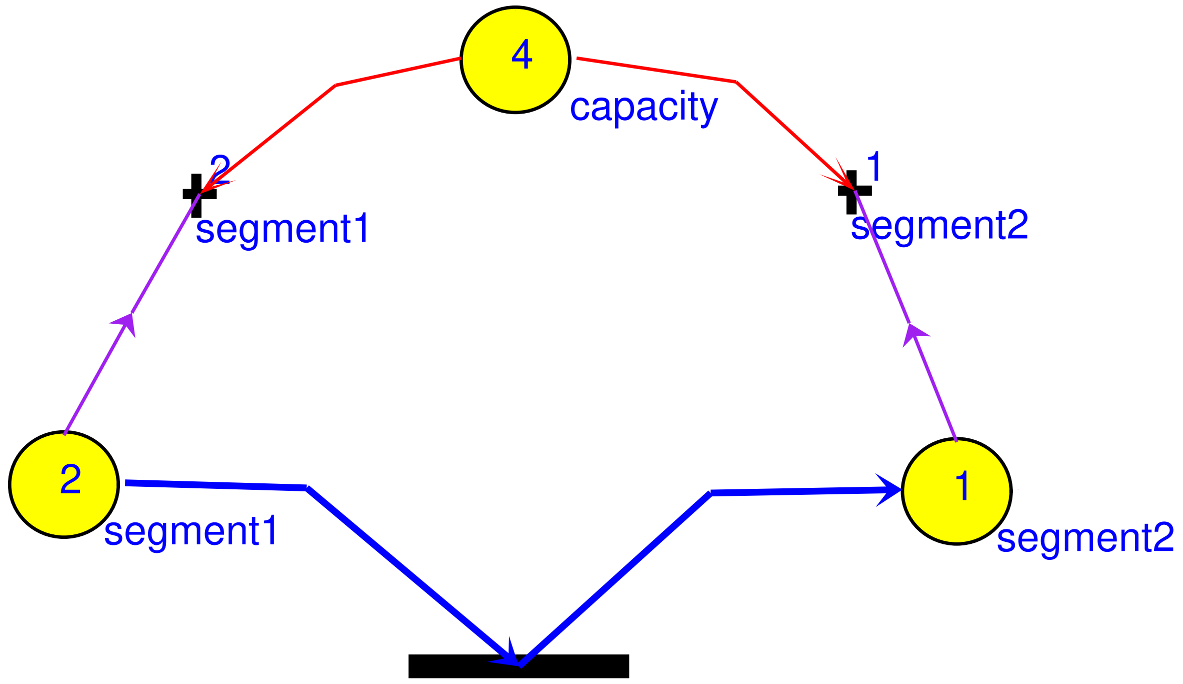
Traffic Capacity to Petri Net Place (links)



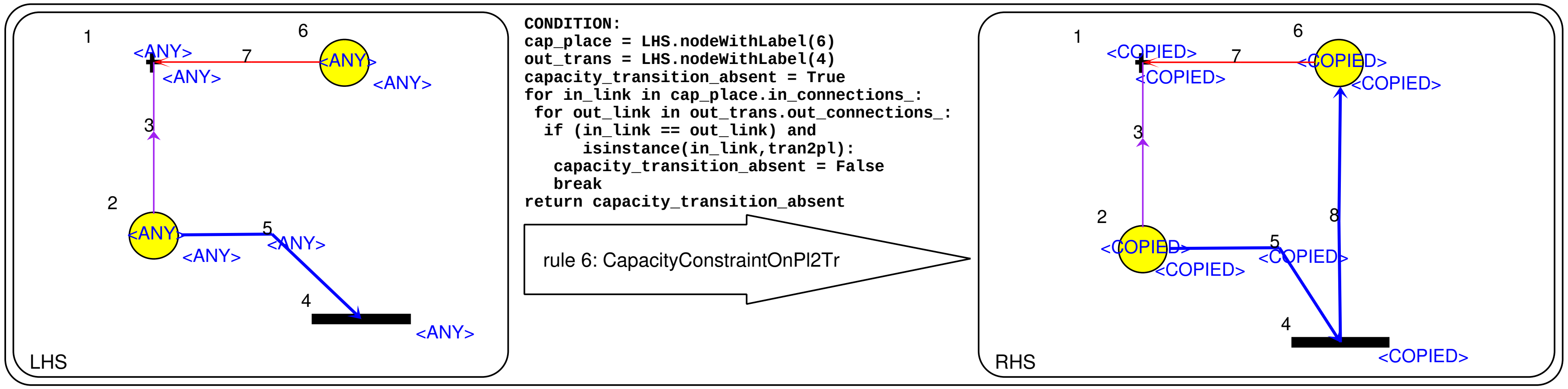
Traffic to Petri Net Graph Transformation rules



Traffic Capacity to Petri Net Place cleanup

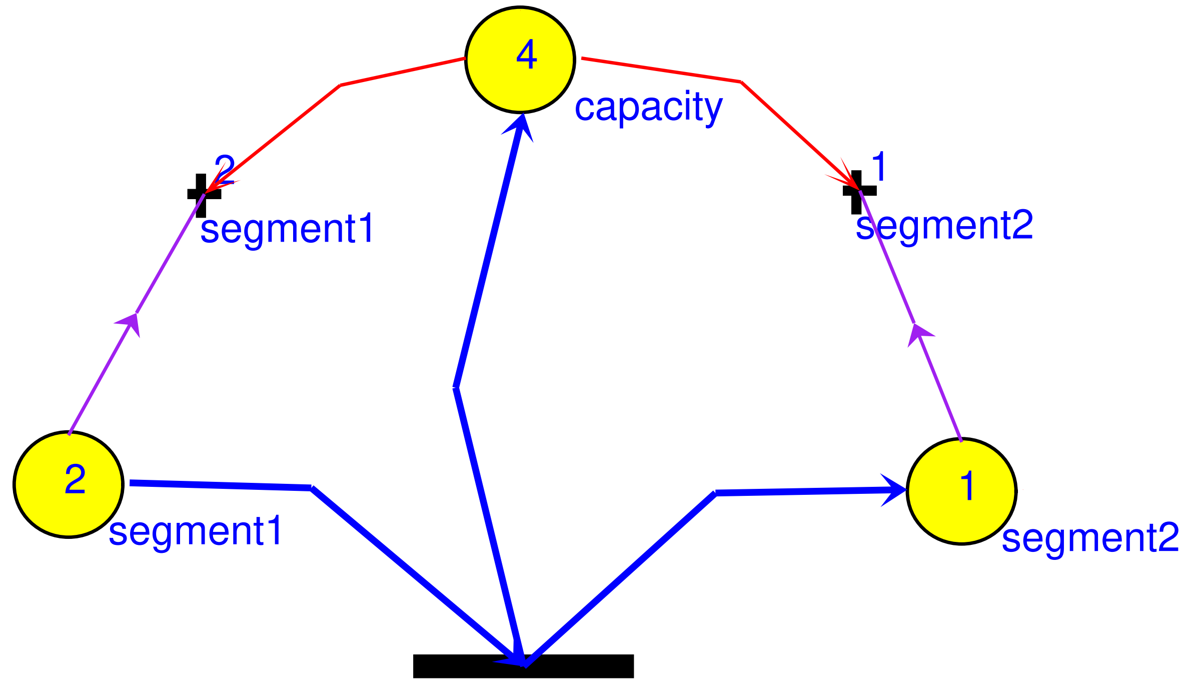


Traffic to Petri Net Graph Transformation rules

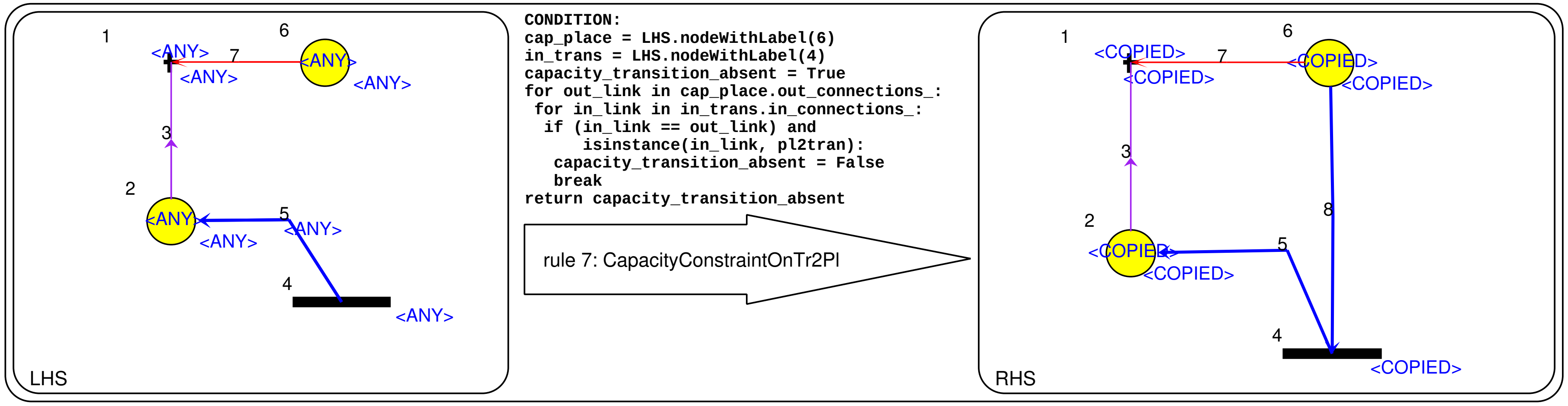


```
CONDITION:
cap_place = LHS.nodeWithLabel(6)
out_trans = LHS.nodeWithLabel(4)
capacity_transition_absent = True
for in_link in cap_place.in_connections_:
 for out_link in out_trans.out_connections_:
  if (in_link == out_link) and
      isinstance(in_link,tran2pl):
   capacity_transition_absent = False
   break
return capacity_transition_absent
```

Capacity Constraint on Place to Transition

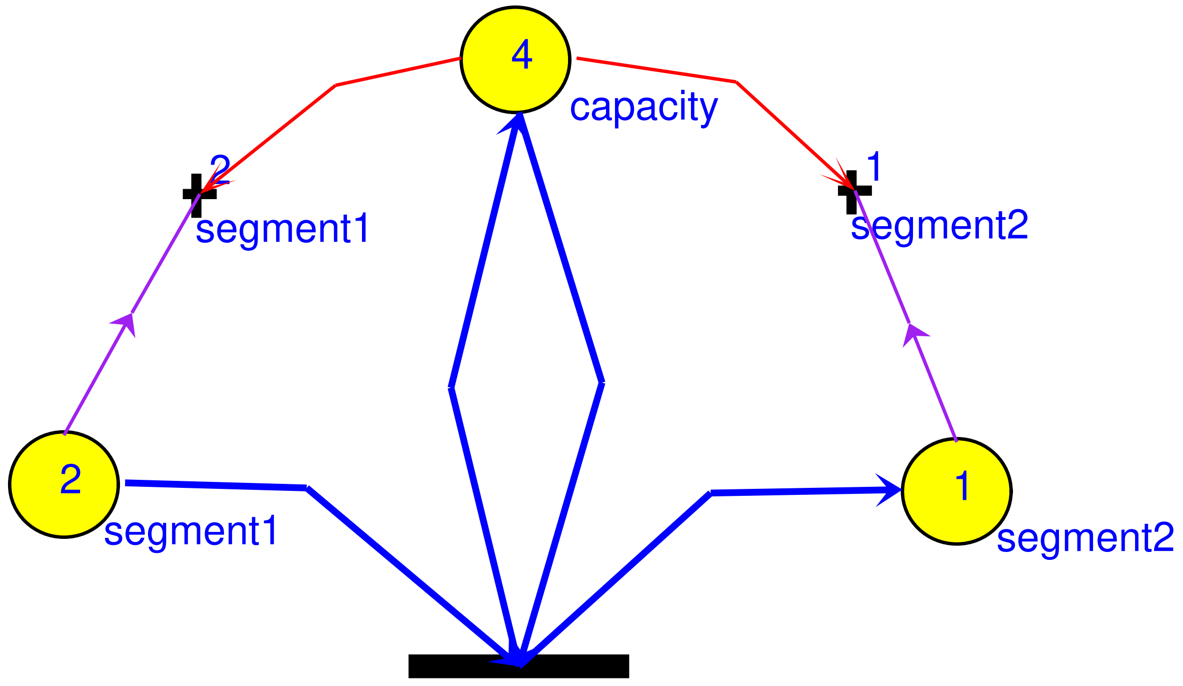


Traffic to Petri Net Graph Transformation rules

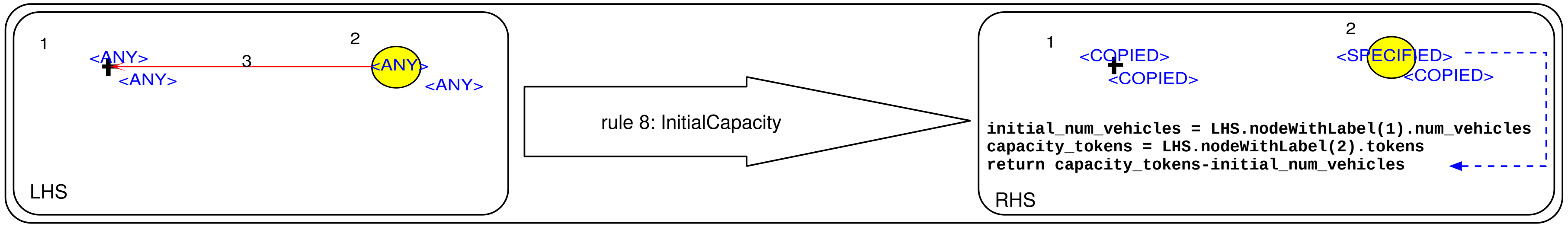


```
CONDITION:
cap_place = LHS.nodeWithLabel(6)
in_trans = LHS.nodeWithLabel(4)
capacity_transition_absent = True
for out_link in cap_place.out_connections_:
 for in_link in in_trans.in_connections_:
  if (in_link == out_link) and
      isinstance(in_link, pl2tran):
   capacity_transition_absent = False
   break
return capacity_transition_absent
```

Capacity Constraint on Transition to Place

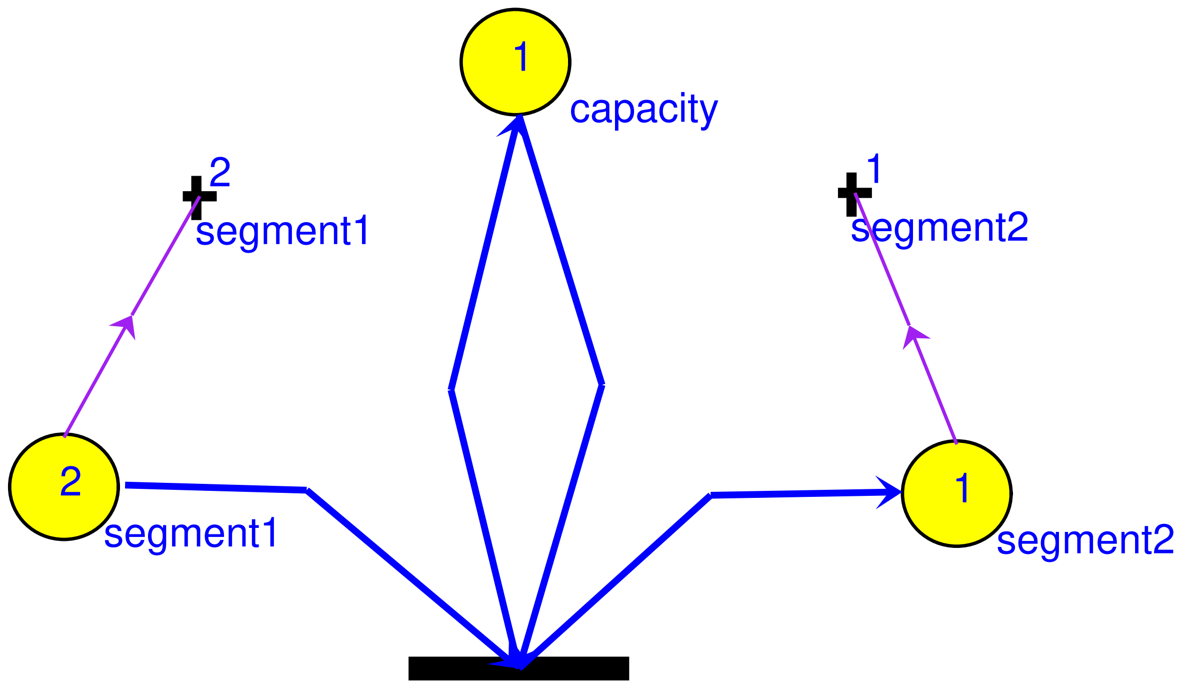


Traffic to Petri Net Graph Transformation rules

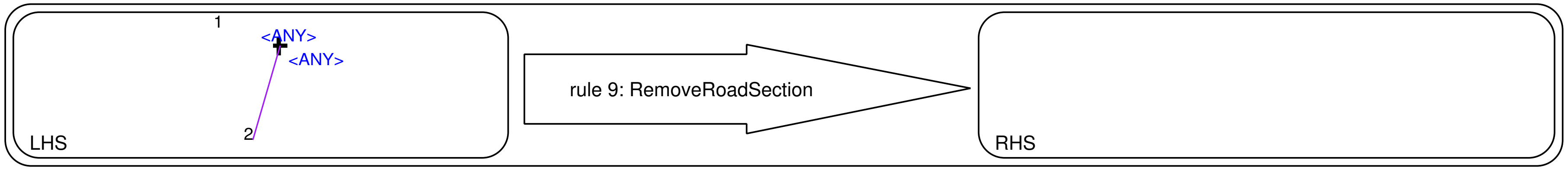


```
initial_num_vehicles = LHS.nodeWithLabel(1).num_vehicles
capacity_tokens = LHS.nodeWithLabel(2).tokens
return capacity_tokens-initial_num_vehicles
```

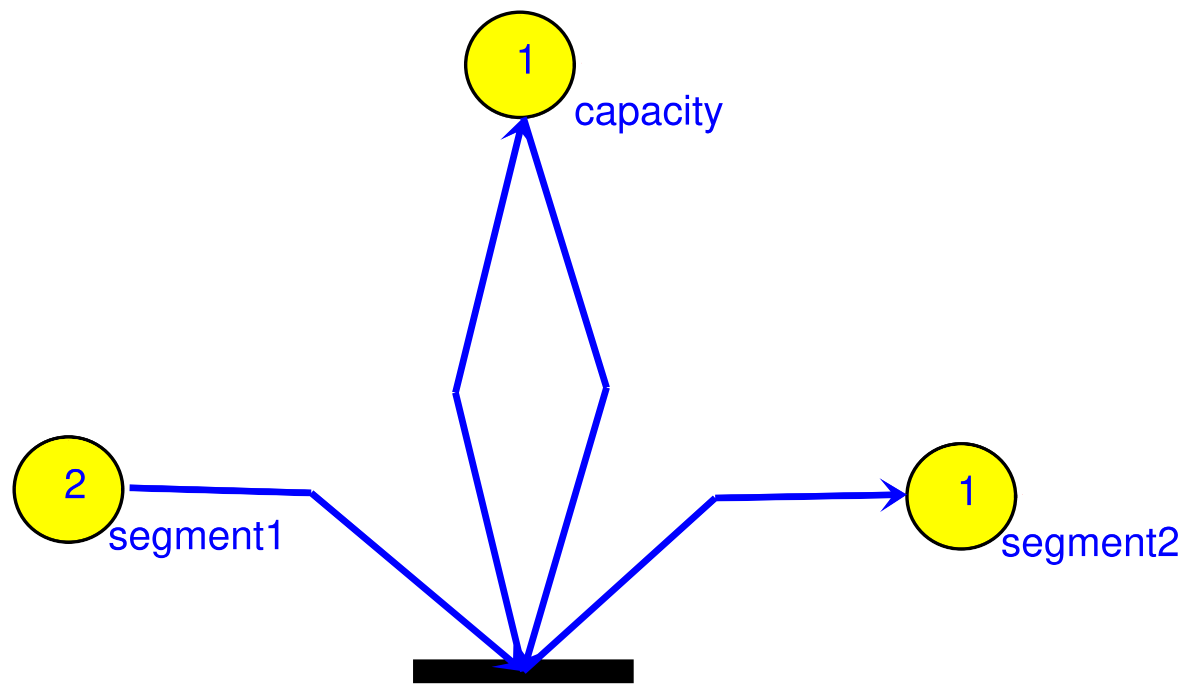
Model Initial Capacity (applied rule twice)



Traffic to Petri Net Graph Transformation rules



Removed Traffic Road Section, now only Petri Net



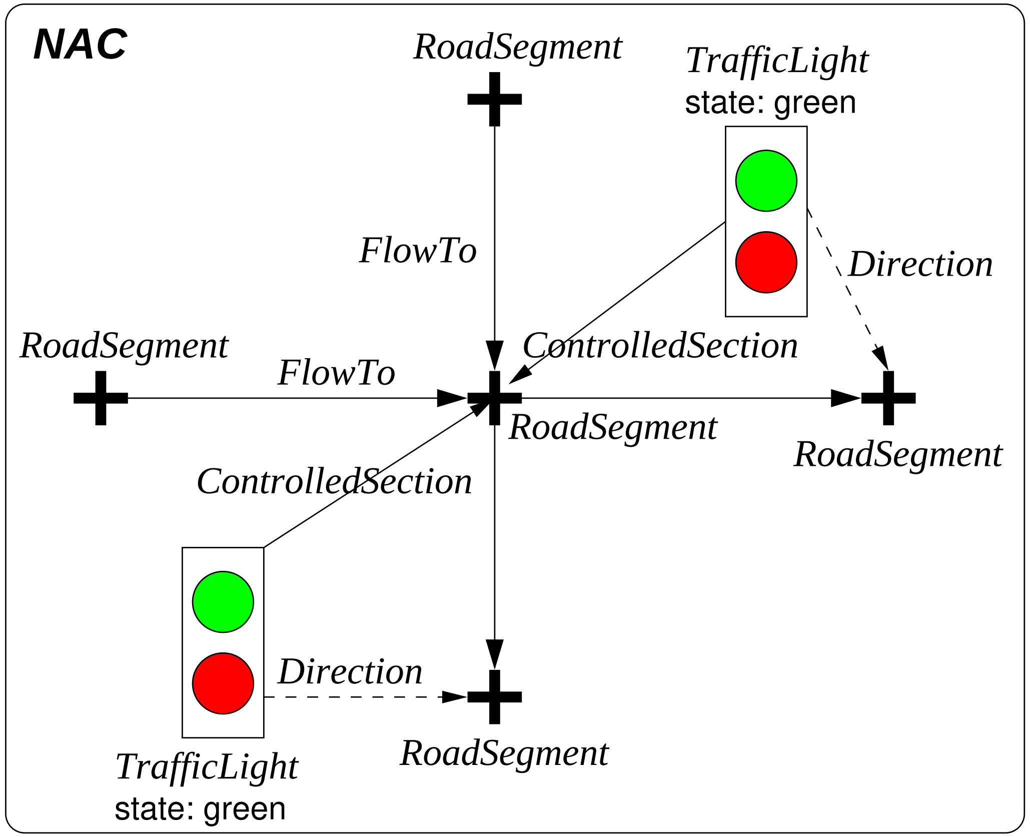
Static Analysis of the Transformation Model

The transformation specified by the Graph Transformation model must satisfy the following requirements:

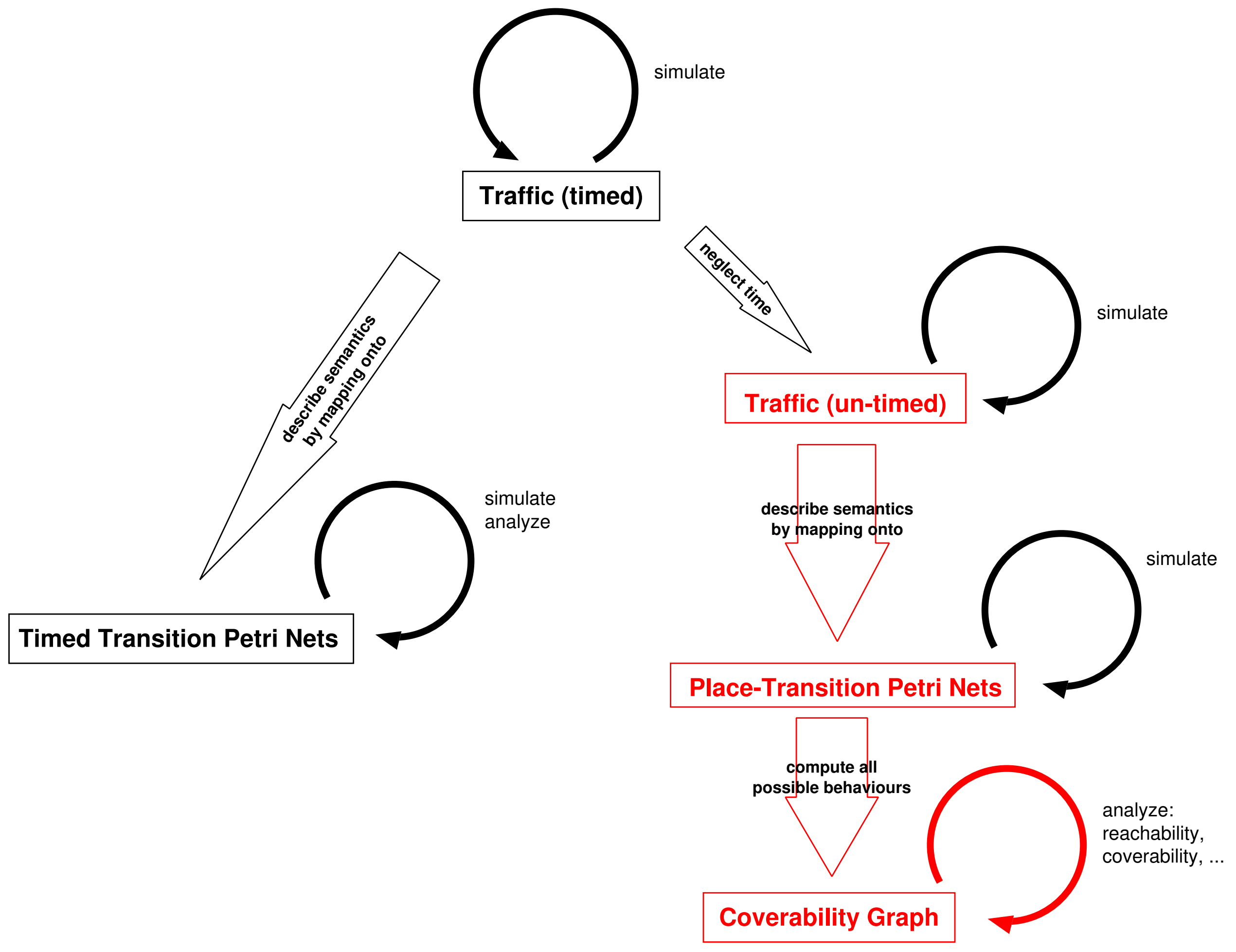
- **Convergence:**
  the transformation process is *finite*
- **Uniqueness:**
  the transformation results in a *single* target model
- **Syntactic Consistency:**
  the target model must be *exclusively* in the target formalism

These properties can often (but not always) be **statically** checked/proved.

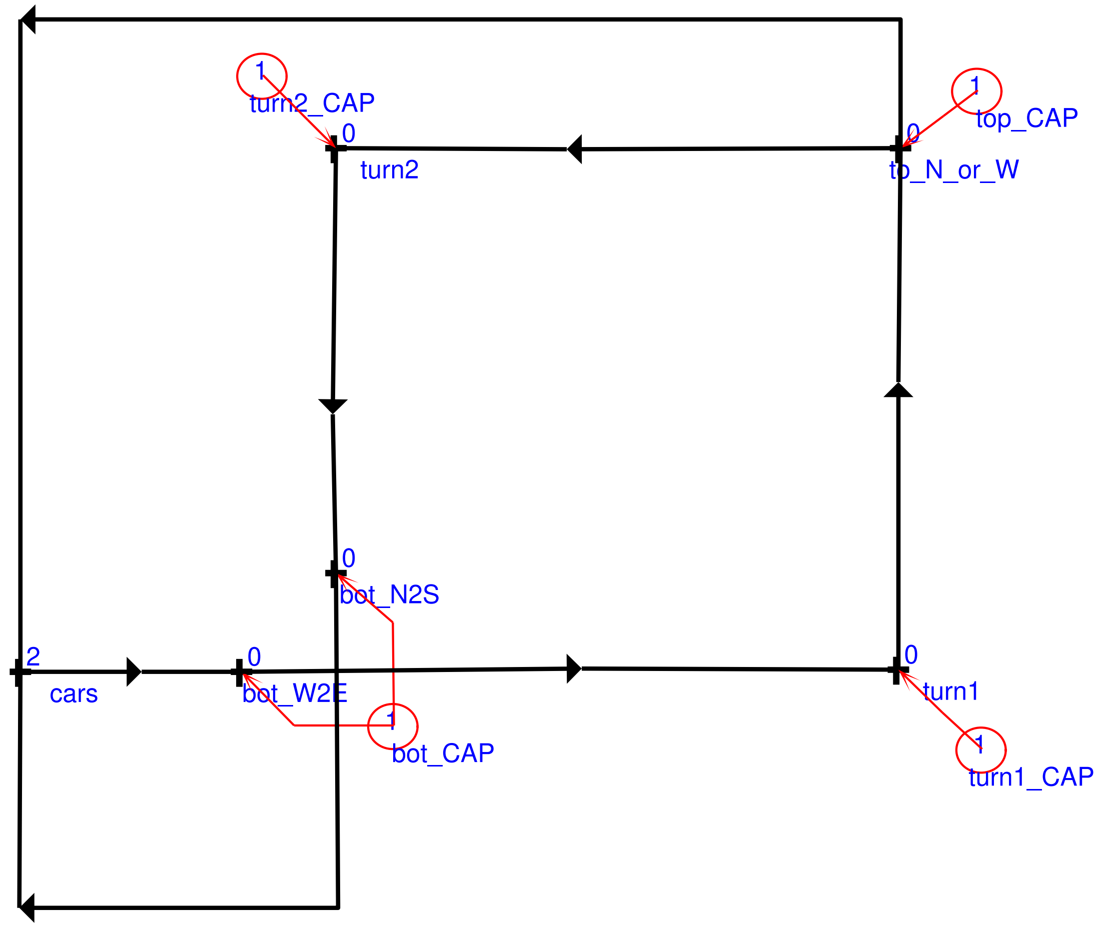
Constraints on Behaviour can be Guaranteed



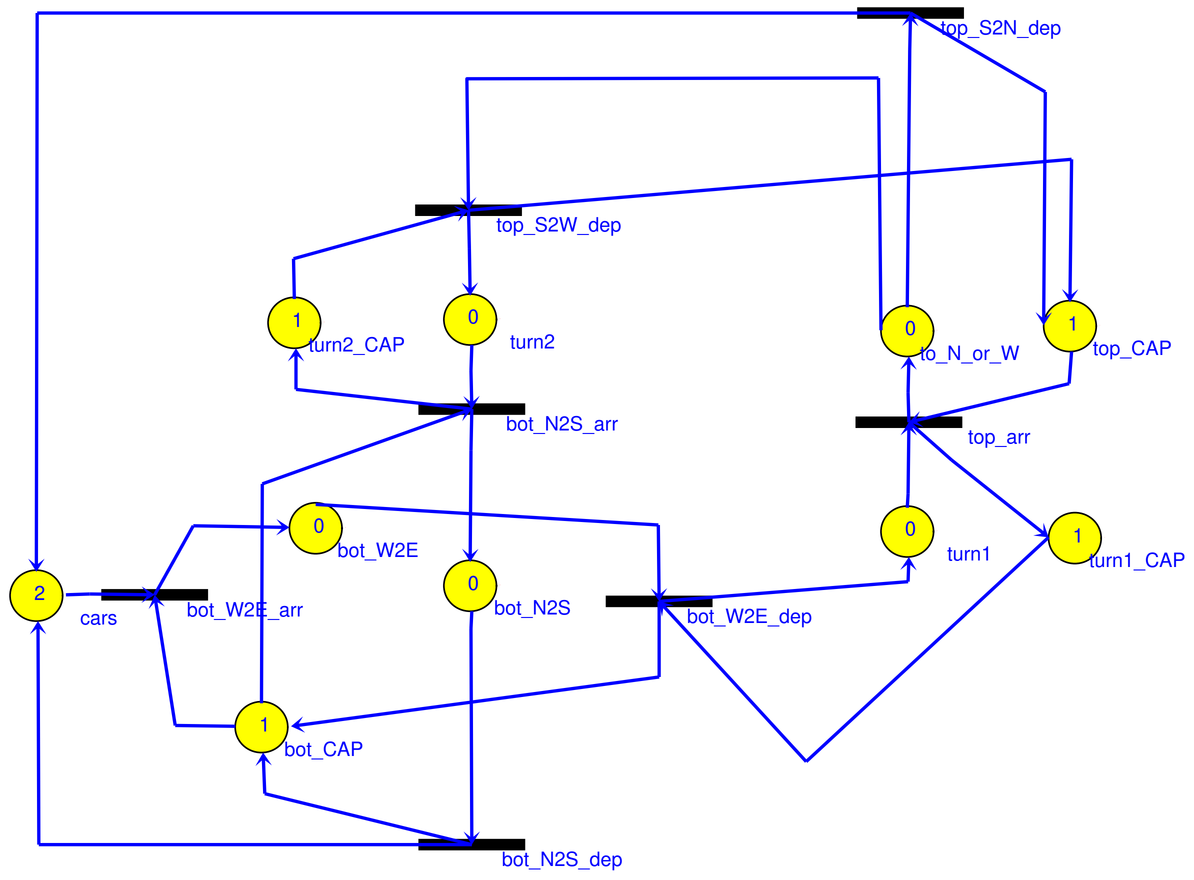
Un-timed Analysis


simulate
Traffic (timed)
describe semantics by mapping onto
neglect time
simulate
Traffic (un-timed)
simulate
analyze
Timed Transition Petri Nets
describe semantics by mapping onto
simulate
Place-Transition Petri Nets
compute all possible behaviours
analyze: reachability, coverability, ...
Coverability Graph

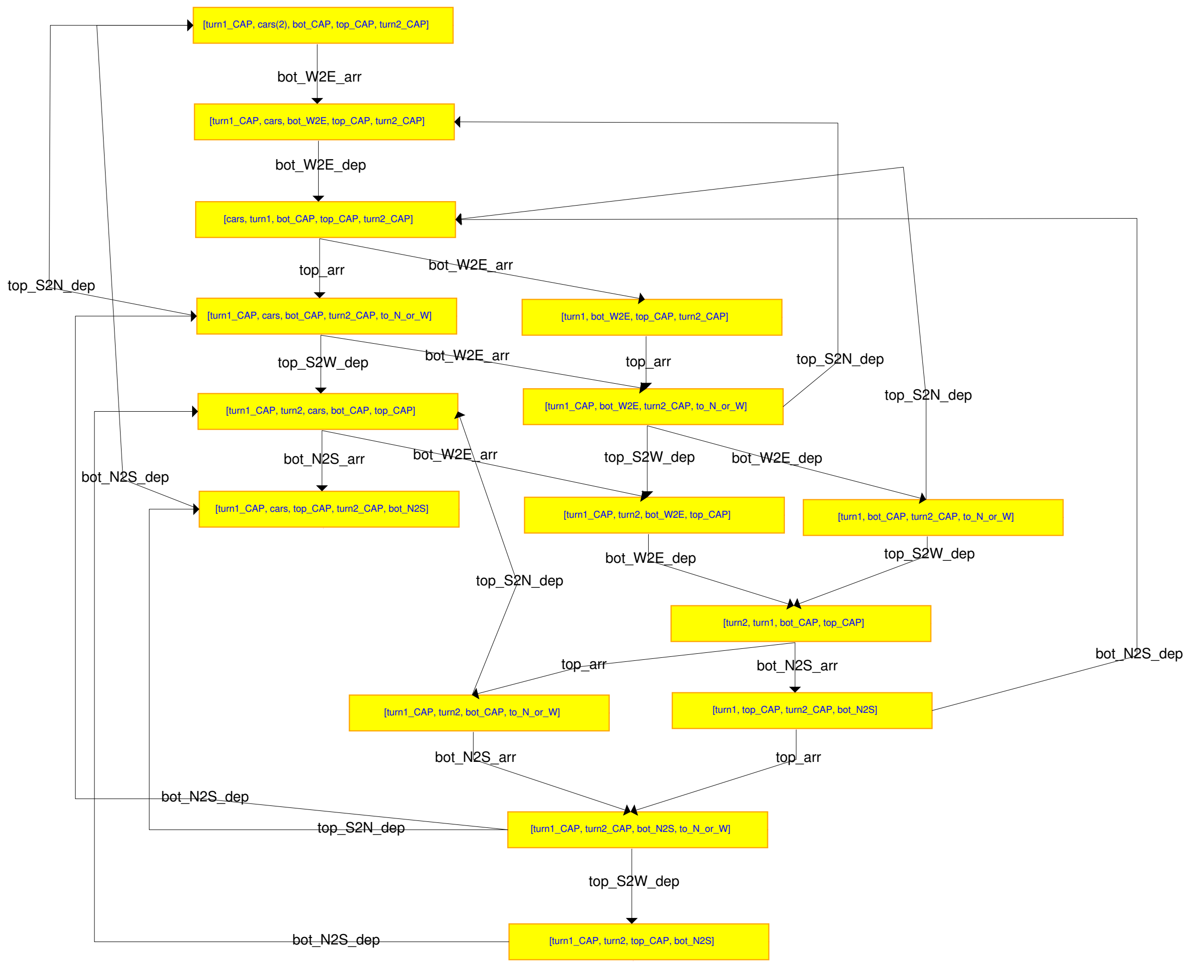
An un-timed Traffic model



the Petri Net describing its behaviour obtained by Graph Rewriting



Analysis: a Coverability Graph for the Petri Net

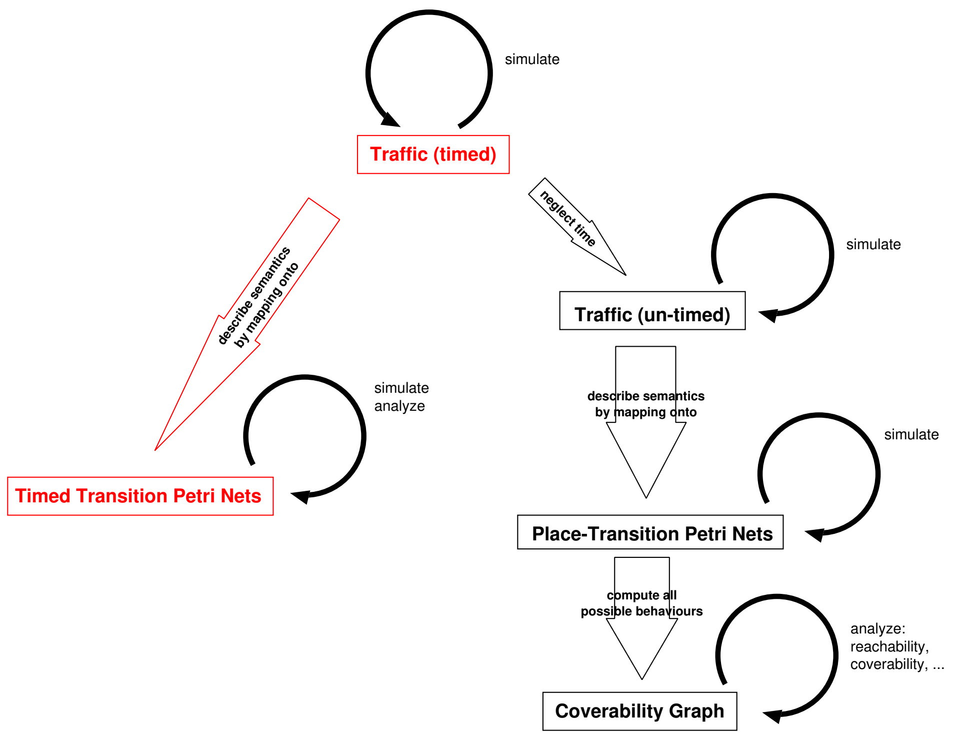


[turn1_CAP, cars(2), bot_CAP, top_CAP, turn2_CAP]
bot_W2E_arr
[turn1_CAP, cars, bot_W2E, top_CAP, turn2_CAP]
bot_W2E_dep
[cars, turn1, bot_CAP, top_CAP, turn2_CAP]
top_arr
bot_W2E_arr
top_S2N_dep
[turn1_CAP, cars, bot_CAP, turn2_CAP, to_N_or_W]
[turn1, bot_W2E, top_CAP, turn2_CAP]
top_S2W_dep
bot_W2E_arr
top_arr
top_S2N_dep
top_S2N_dep
[turn1_CAP, turn2, cars, bot_CAP, top_CAP]
[turn1_CAP, bot_W2E, turn2_CAP, to_N_or_W]
bot_N2S_arr
bot_W2E_arr
top_S2W_dep
bot_W2E_dep
bot_N2S_dep
[turn1_CAP, cars, top_CAP, turn2_CAP, bot_N2S]
[turn1_CAP, turn2, bot_W2E, top_CAP]
[turn1, bot_CAP, turn2_CAP, to_N_or_W]
bot_W2E_dep
top_S2W_dep
top_S2N_dep
[turn2, turn1, bot_CAP, top_CAP]
top_arr
bot_N2S_arr
bot_N2S_dep
[turn1_CAP, turn2, bot_CAP, to_N_or_W]
[turn1, top_CAP, turn2_CAP, bot_N2S]
bot_N2S_arr
top_arr
bot_N2S_dep
top_S2N_dep
[turn1_CAP, turn2_CAP, bot_N2S, to_N_or_W]
top_S2W_dep
bot_N2S_dep
[turn1_CAP, turn2, top_CAP, bot_N2S]

Conservation Analysis

```
1.0 x[turn1_CAP] + 1.0 x[turn1] = 1.0

1.0 x[cars] + 1.0 x[bot_W2E] + 1.0 x[turn1] +
1.0 x[to_N_or_W] + 1.0 x[turn2] + 1.0 x[bot_N2S] = 2.0

1.0 x[top_CAP] + 1.0 x[to_N_or_W] = 1.0

1.0 x[turn2_CAP] + 1.0 x[turn2] = 1.0

1.0 x[bot_CAP] + 1.0 x[bot_W2E] + 1.0 x[bot_N2S] = 1.0
```

Timed Semantics by mapping onto TTPN



simulate
Traffic (timed)
neglect time
simulate
describe semantics by mapping onto
Traffic (un-timed)
simulate analyze
describe semantics by mapping onto
simulate
Timed Transition Petri Nets
Place-Transition Petri Nets
compute all possible behaviours
analyze: reachability, coverability, ...
Coverability Graph

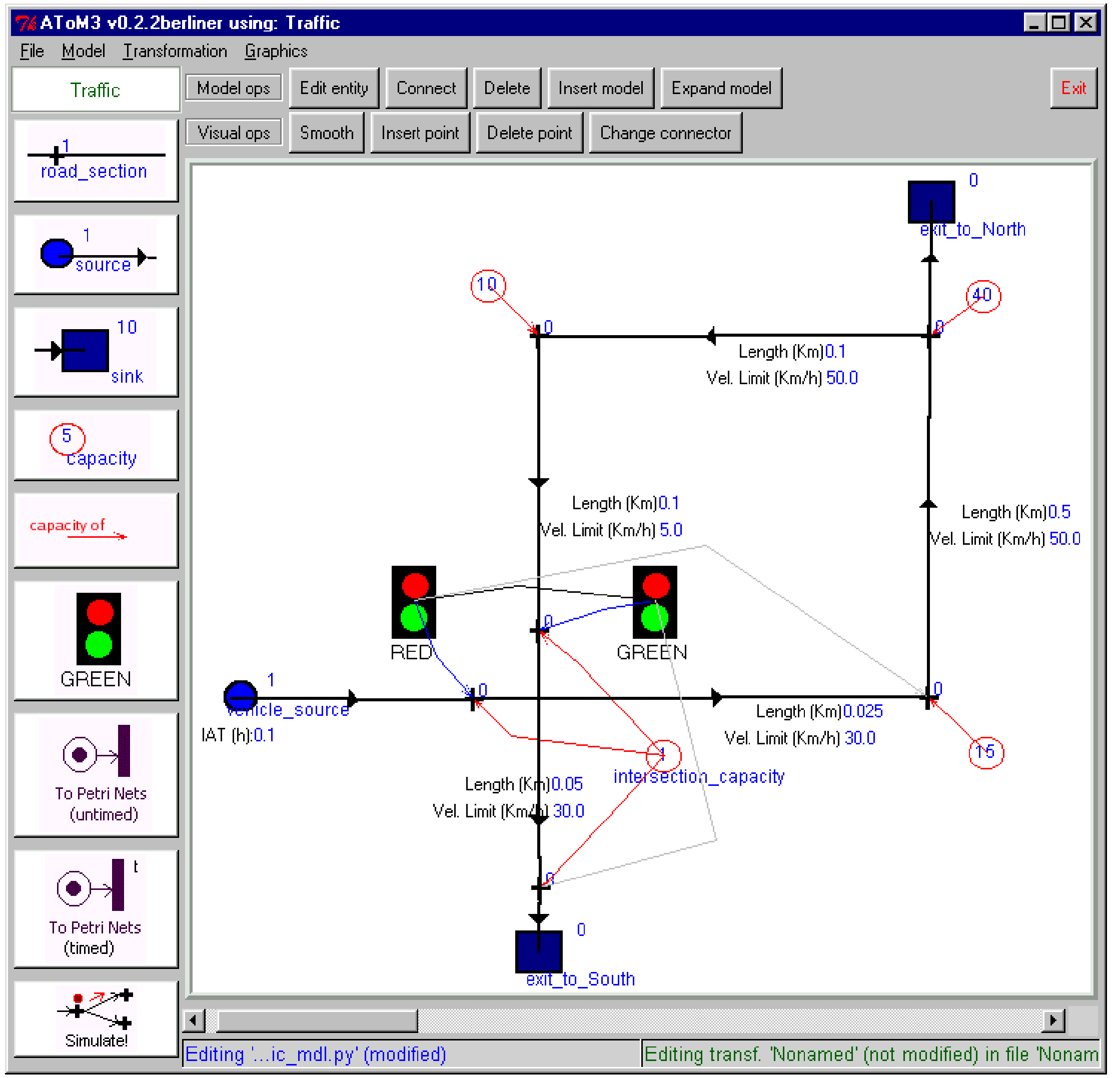
Juan de Lara, Hans Vangheluwe, and Pieter J. Mosterman.

Modelling and analysis of traffic networks based on graph transformation.

Formal Methods for Automation and Safety in Railway and Automotive Systems.

December 2004. Braunschweig, Germany.

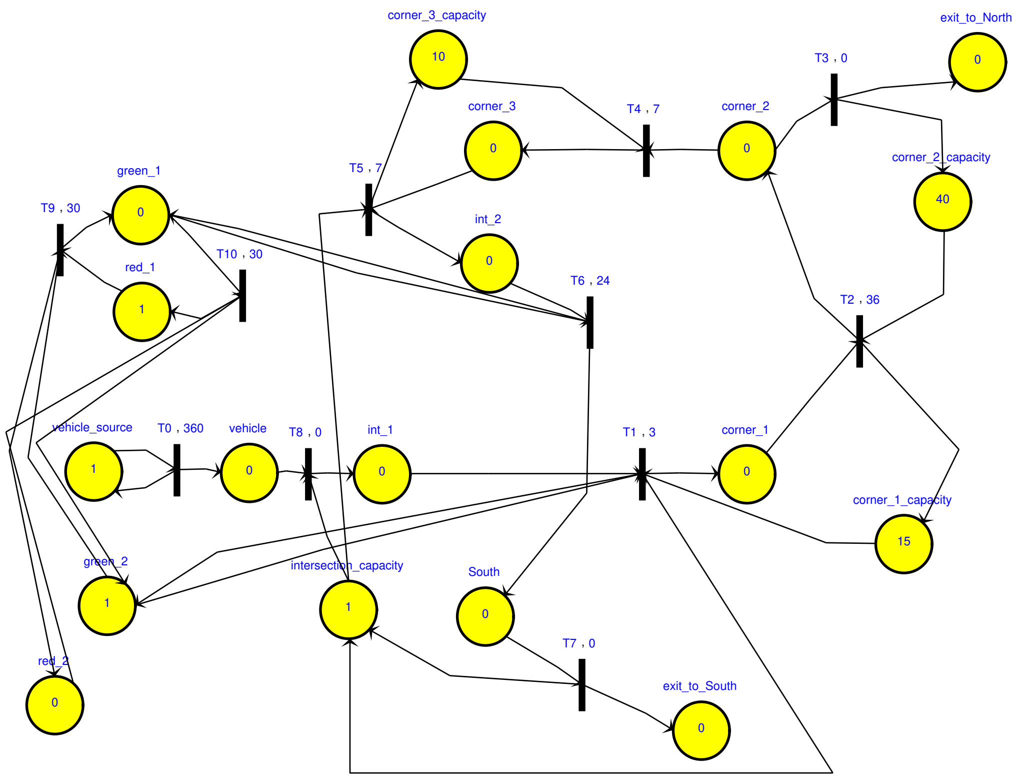
Traffic Network



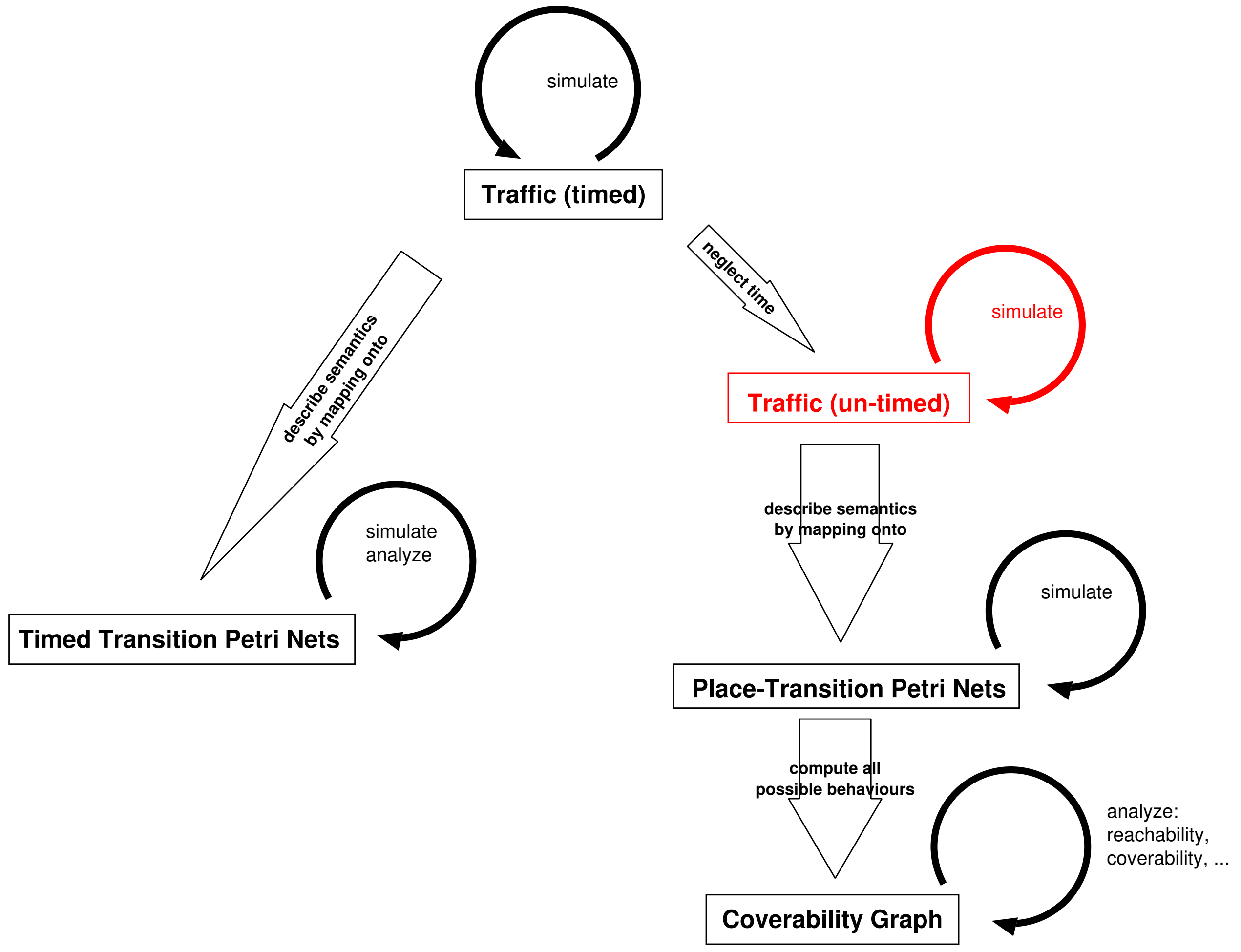
Editing '...ic_mdl.py' (modified)

Editing transf. 'Nonamed' (not modified) in file 'Nonam

Converted to TTPN
(ready for analysis/simulation)



Iterative Simulation (Executing GG)



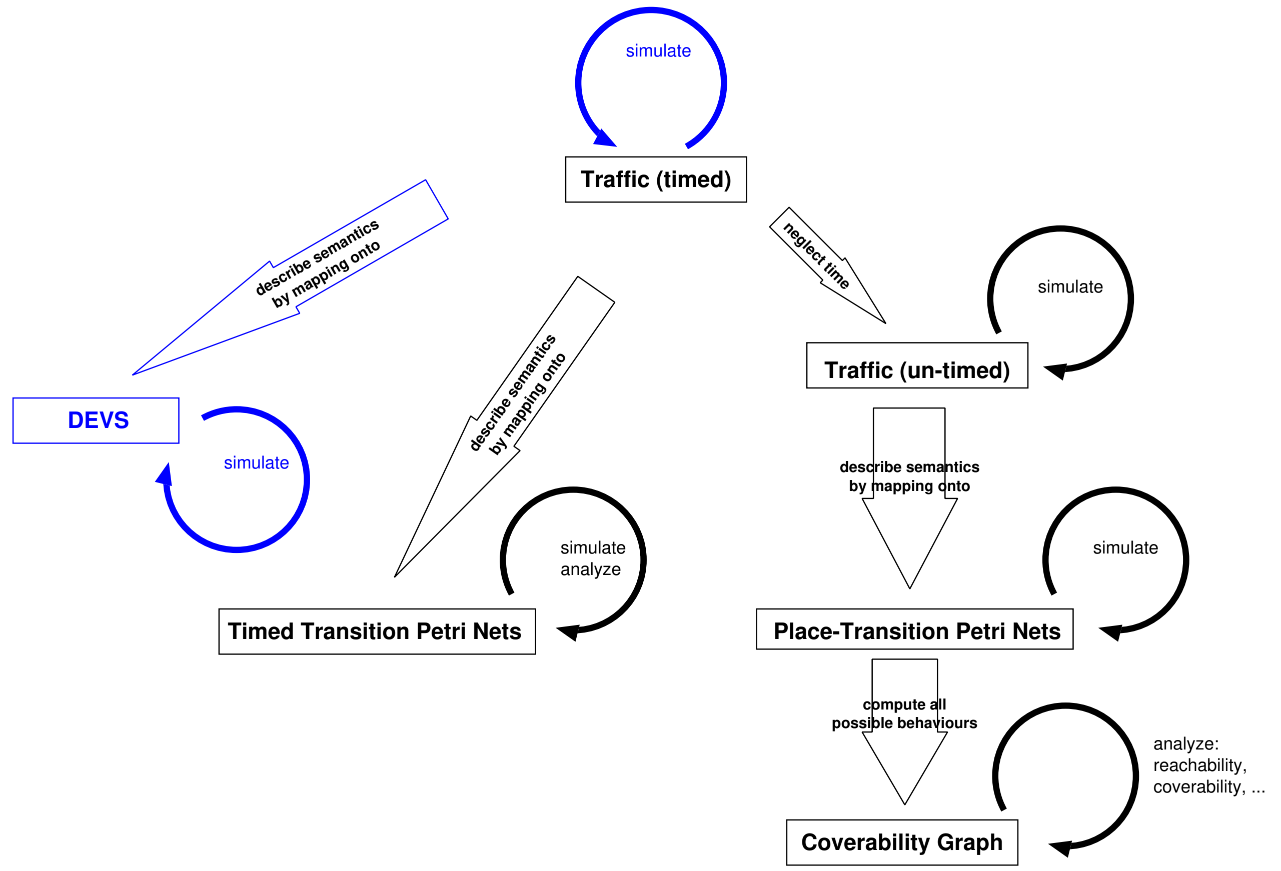
simulate
Traffic (timed)
describe semantics by mapping onto
neglect time
simulate
Traffic (un-timed)
simulate analyze
Timed Transition Petri Nets
describe semantics by mapping onto
simulate
Place-Transition Petri Nets
compute all possible behaviours
analyze: reachability, coverability, ...
Coverability Graph

Conclusions

Demonstrated **feasibility** of rapidly and re-usably building Domain Specific Visual Modelling, Analysis, Simulation tools using **meta-modelling** and **graph rewriting**.

model everything

Future Work


simulate
Traffic (timed)
describe semantics by mapping onto
neglect time
simulate
Traffic (un-timed)
DEVS
simulate
describe semantics by mapping onto
describe semantics by mapping onto
simulate analyze
simulate
Timed Transition Petri Nets
Place-Transition Petri Nets
compute all possible behaviours
analyze: reachability, coverability, ...
Coverability Graph

. . . and add hierarchy