

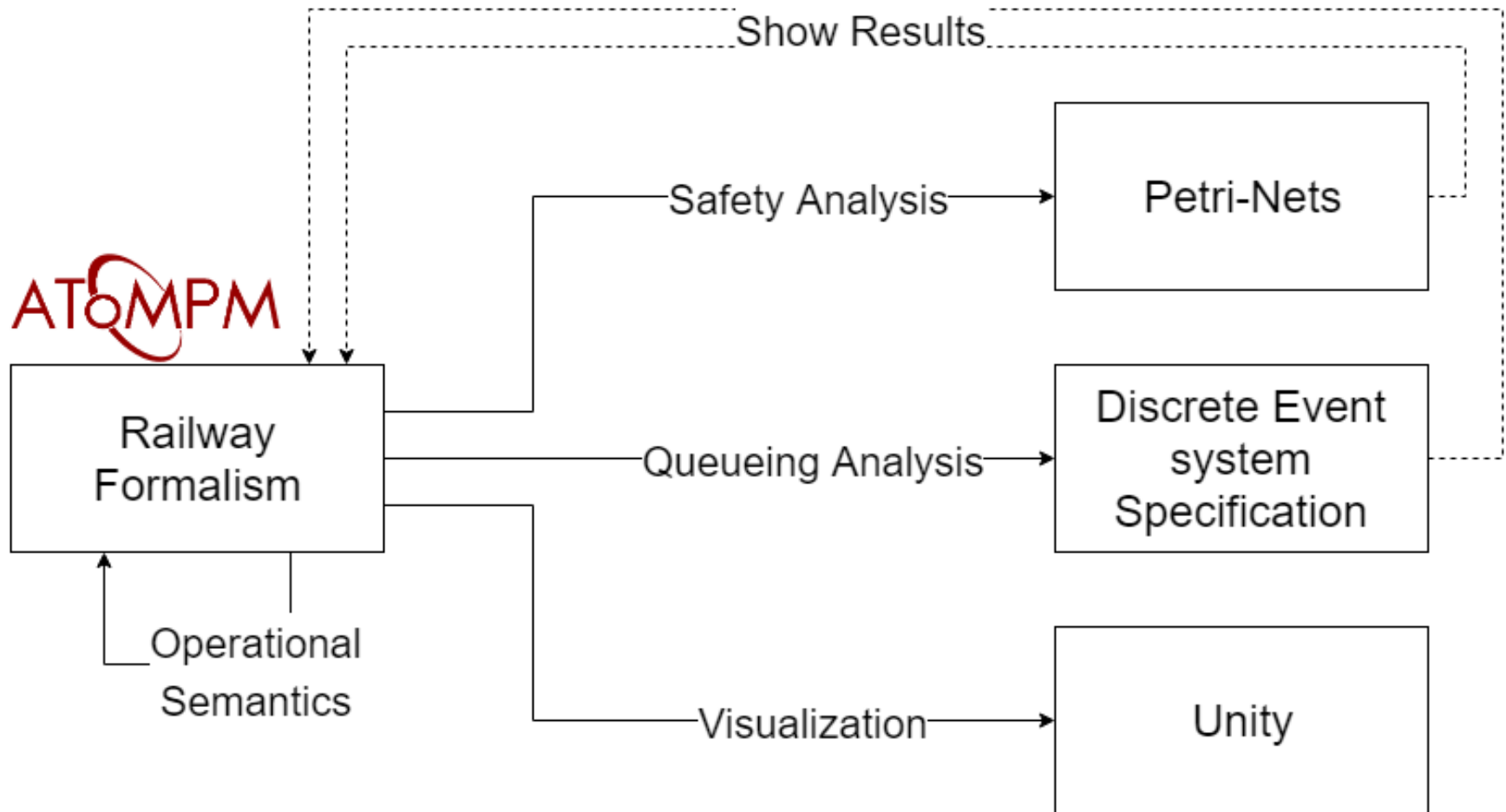
Mapping the Railway formalism onto different domains

Zhong Xi Lu

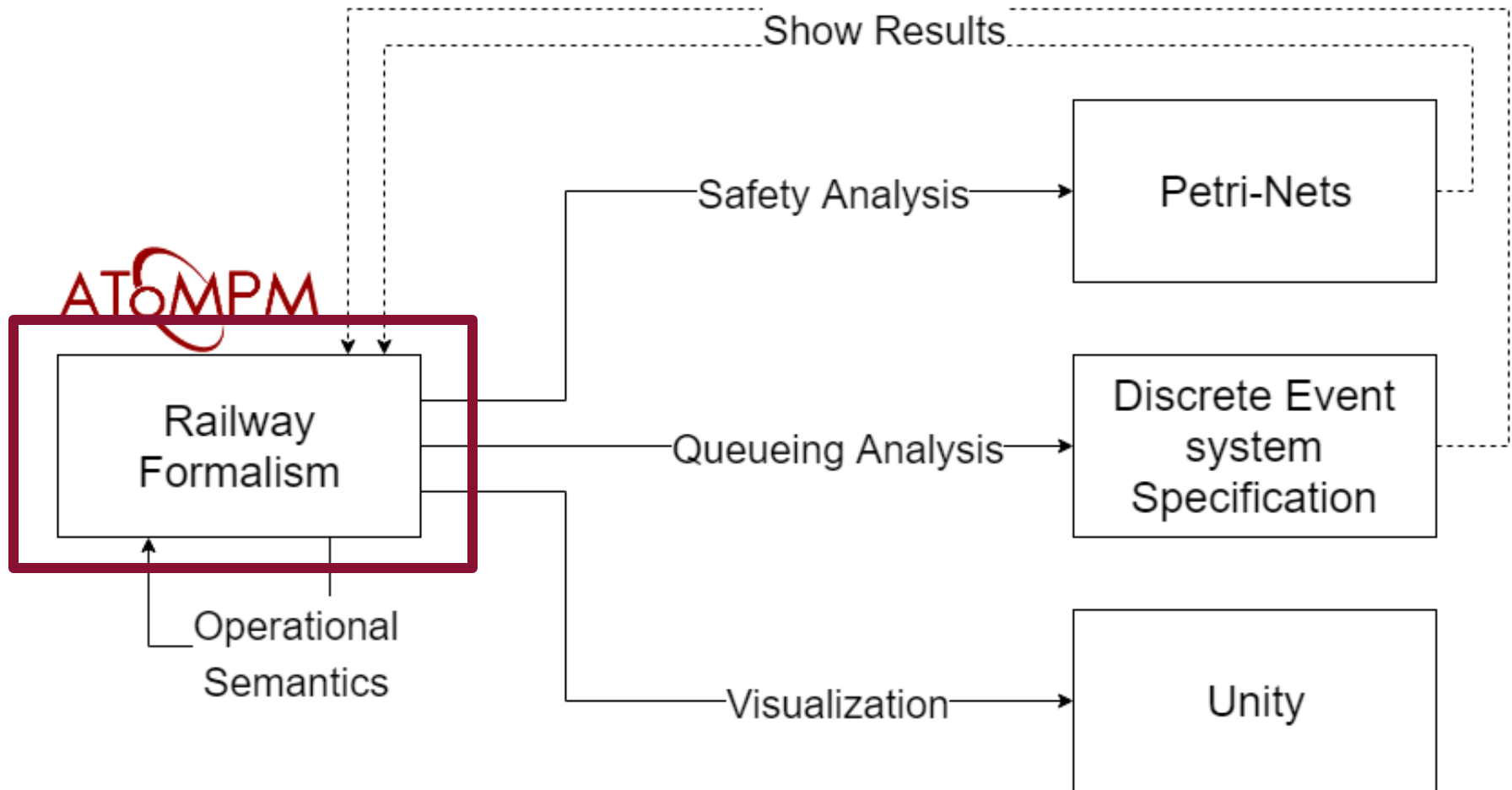
Promoter: Hans Vangheluwe

Supervisor: Simon Van Mierlo

Overview



1. Abstract and Concrete Syntax



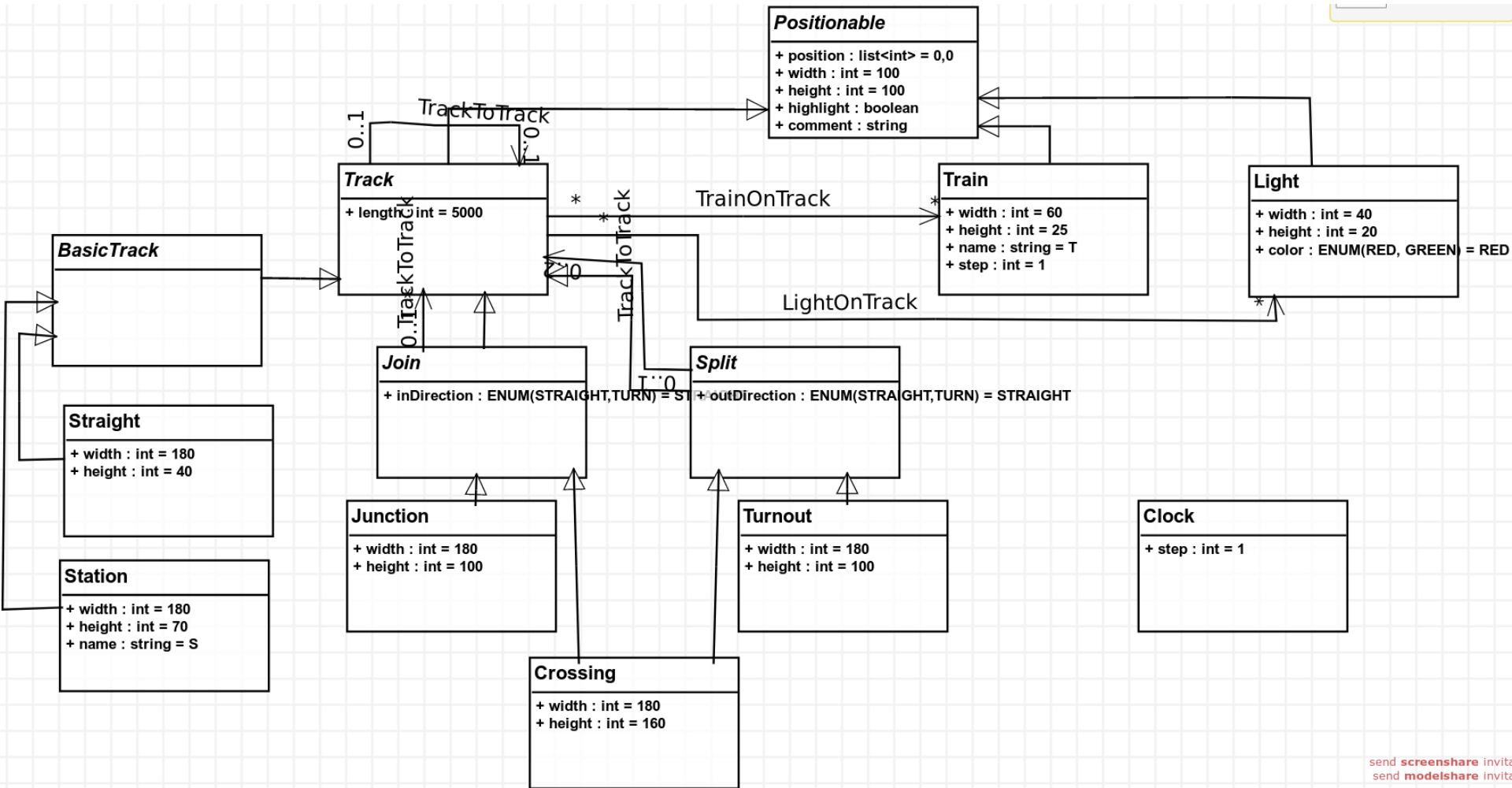
Railway Formalism

- **Tracks**
 - Straight
 - Turnout
 - Junction
 - Crossing
 - Station
- **Trains**
- **Lights**

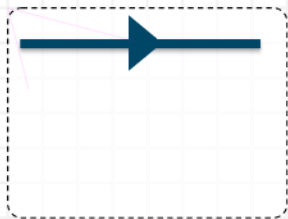
Based on: *Railway Operation and Control* by Joern Pachl



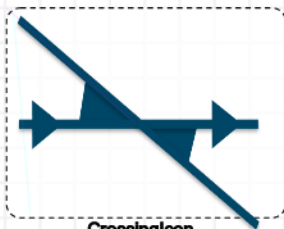
Abstract Syntax



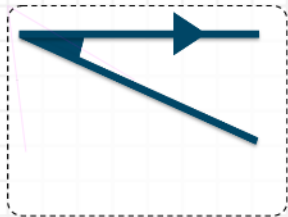
Concrete Syntax



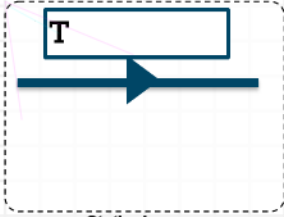
StraightIcon



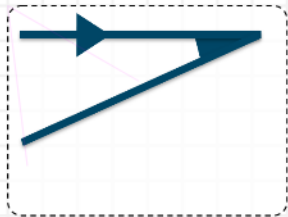
CrossingIcon



TurnoutIcon



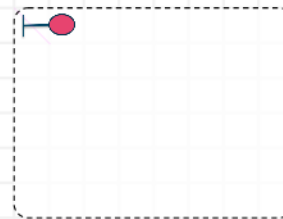
StationIcon



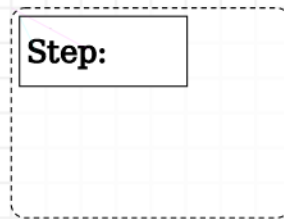
JunctionIcon



TrainIcon



LightIcon



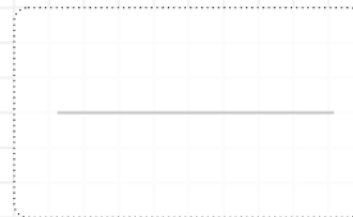
ClockIcon



TrackToTrackLink

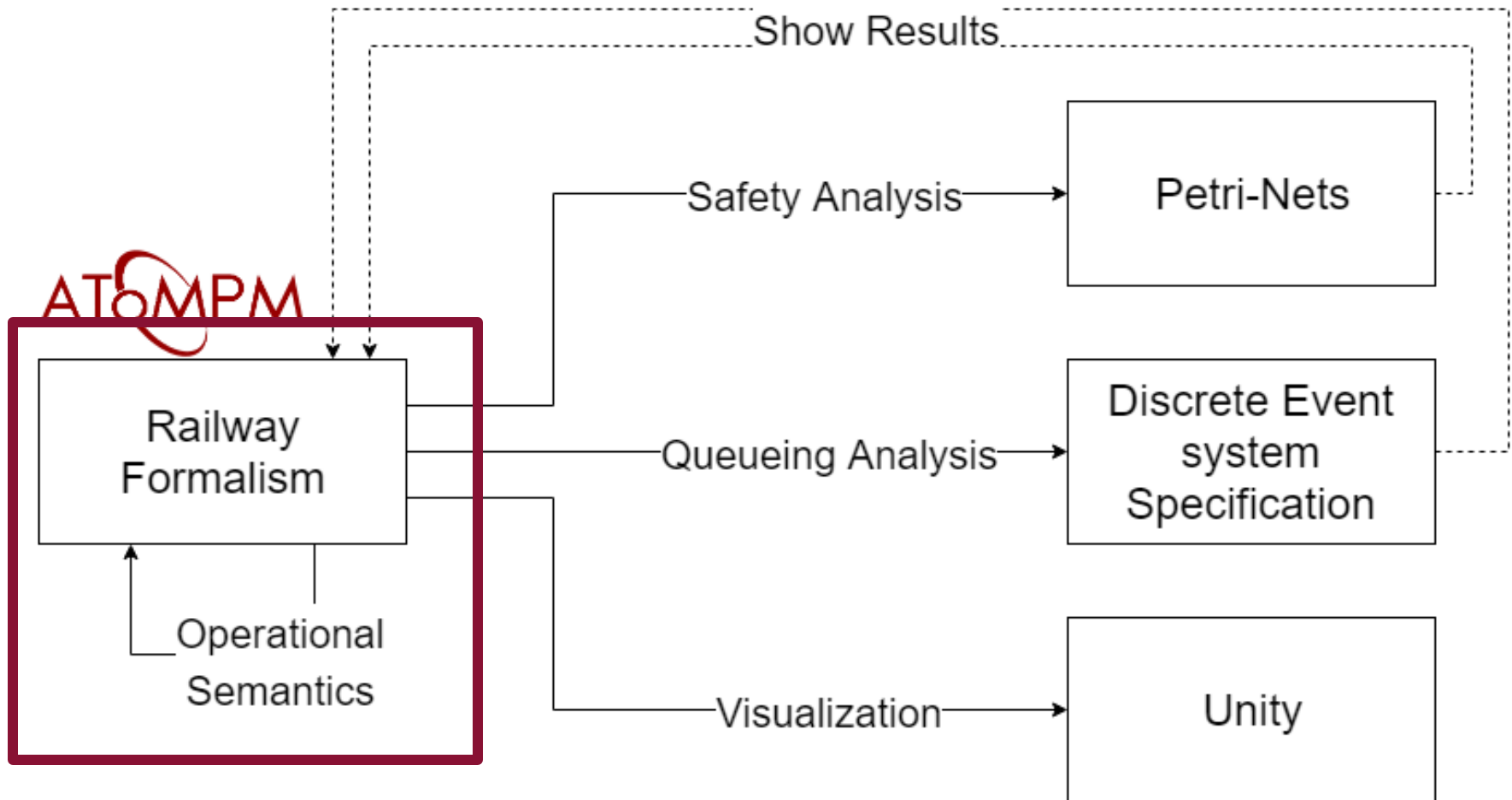


TrainOnTrackLink

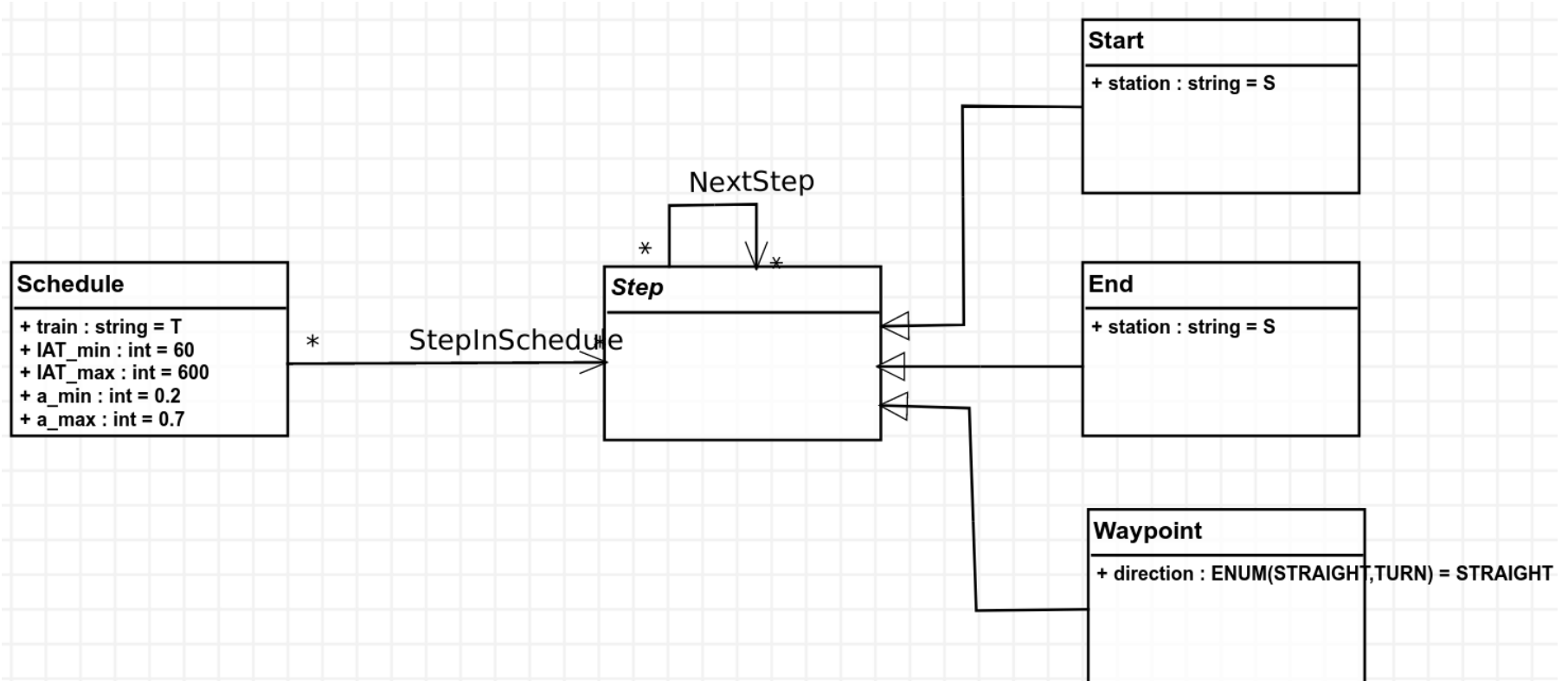


LightOnTrackLink

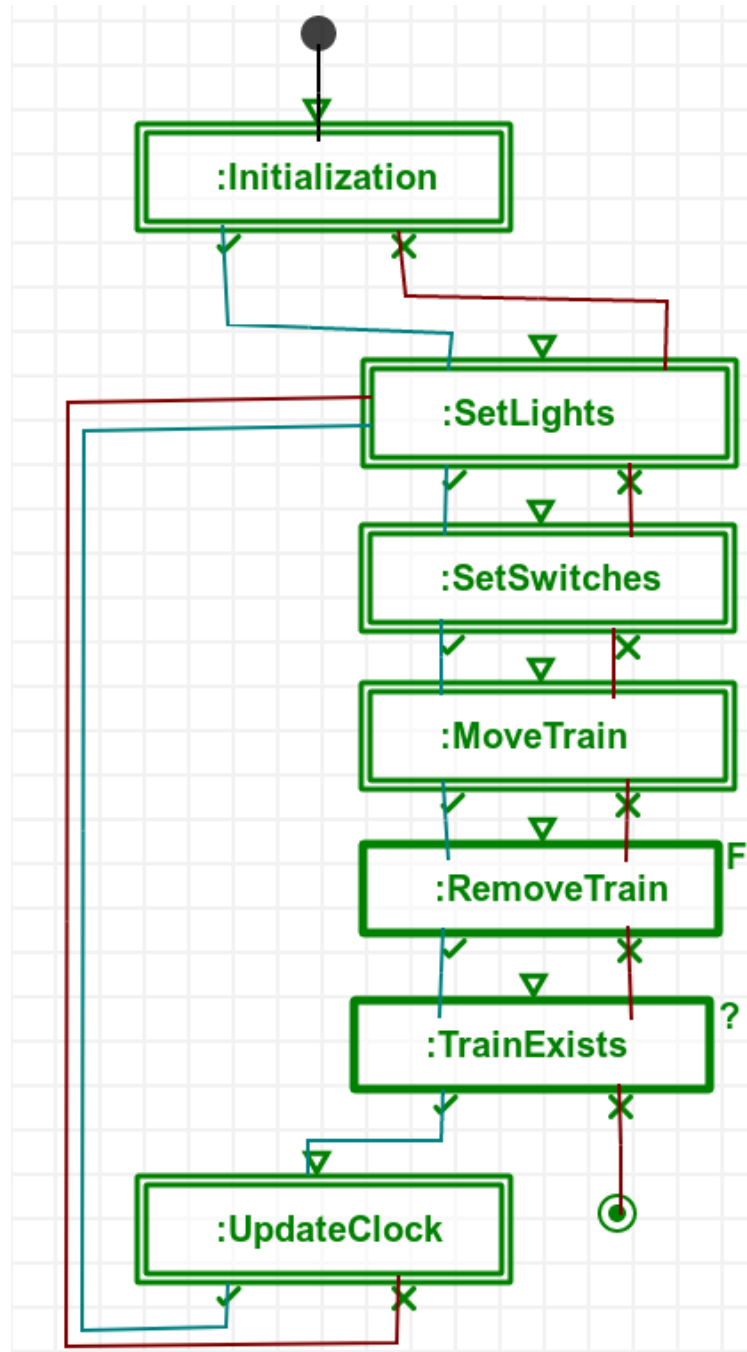
2. Operational Semantics



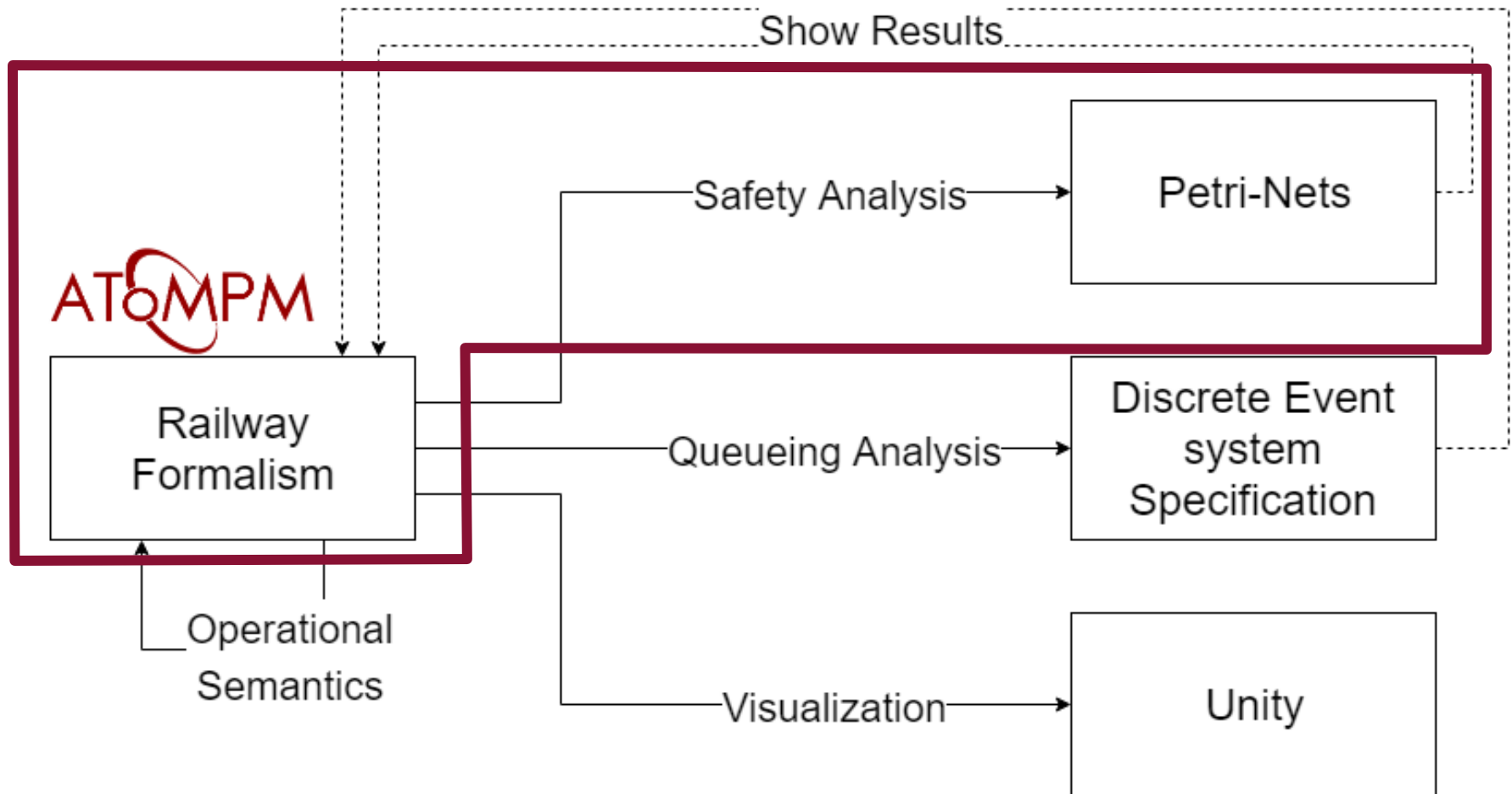
Train Schedule Formalism



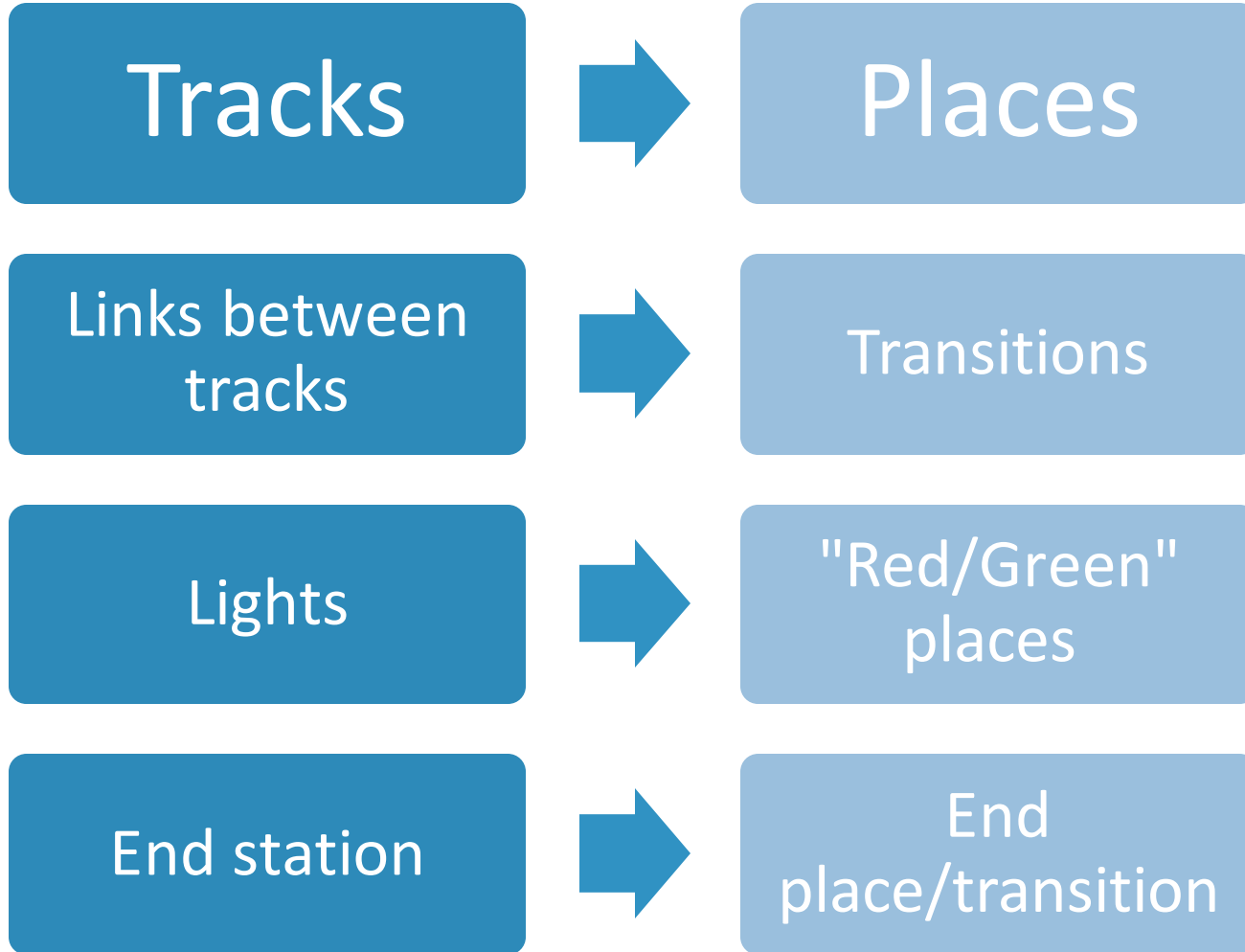
Schedule



3. Safety Analysis



Mapping to Petri-nets



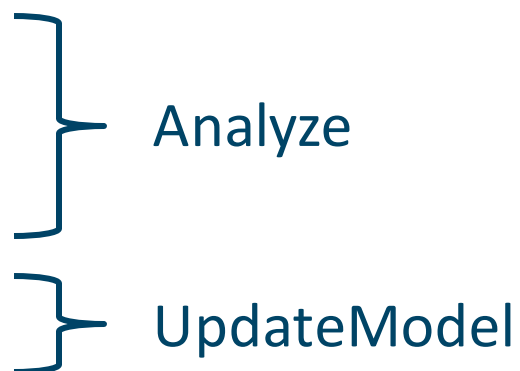
LoLA

- A Low Level Petri net Analyzer
- Command line tool
- Specify custom properties through CTL formulas
(Computation Tree Logic)

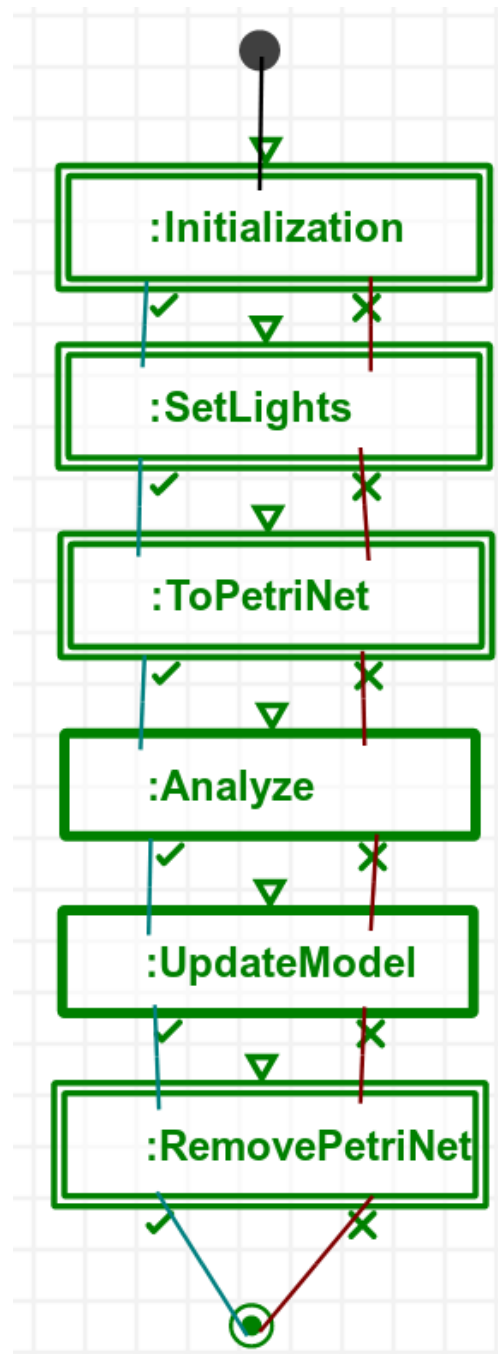
Safety Properties

- Deadlock: $EF \text{ DEADLOCK}$
- Reachability: $EF T > 0$
- Safeness: $AG T \leq 1$
- Lights Invariant: $AG (G = 1 \text{ OR } R = 1)$

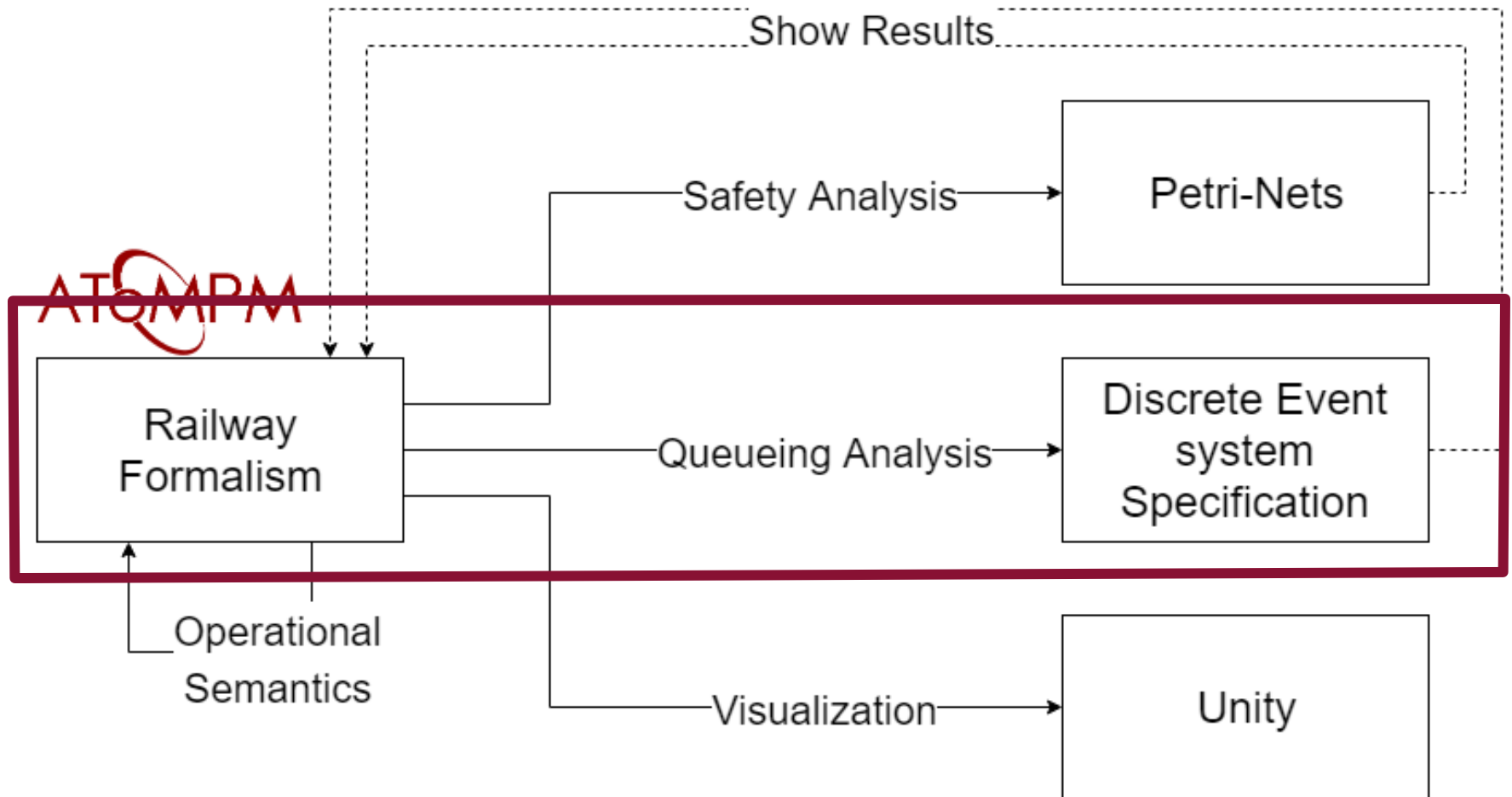
Interfaces

- Use ID's for traceability (\$atompmlId)
 - Generate LoLA petri net file
 - Call LoLA via command
 - Read results from files
- 
- Analyze
- UpdateModel

Schedule



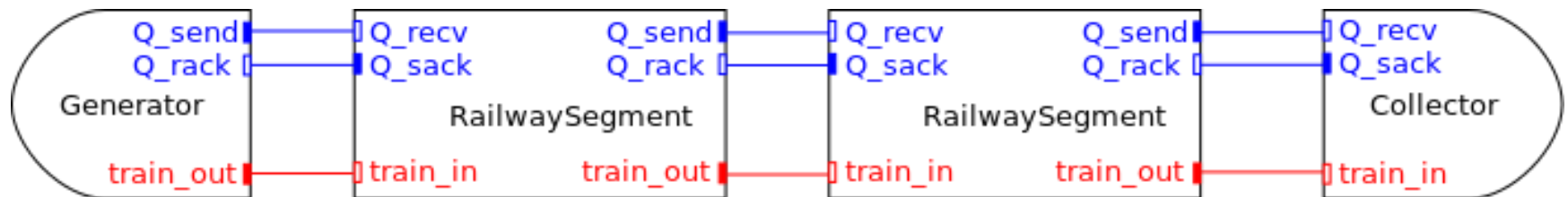
4. Queueing Analysis



DEVS Model

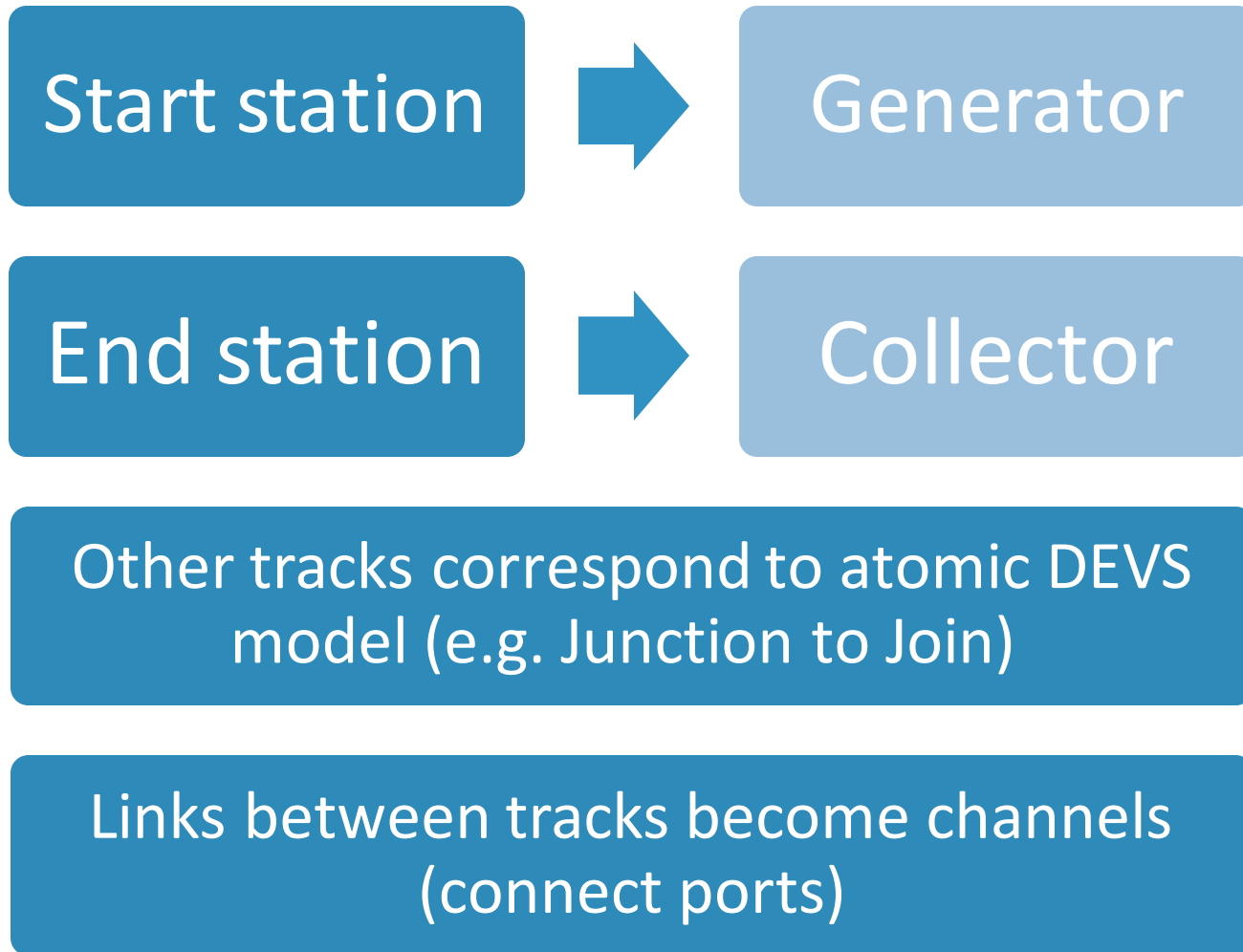
- Using PythonPDEVS
- Atomic models:
 - RailwaySegment
 - Join
 - Split
 - Crossing
 - Generator
 - Collector

DEVS Model Example



Source: <http://msdl.cs.mcgill.ca/people/hv/teaching/MoSIS/assignments/DEVS>

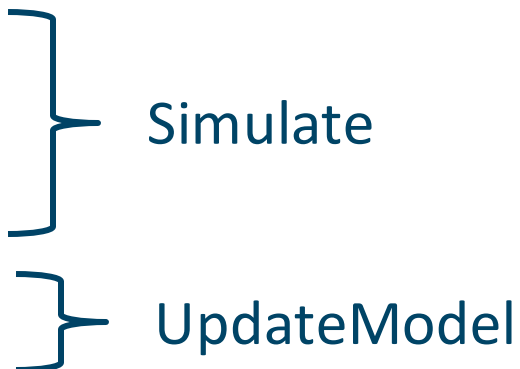
Mapping to DEVS



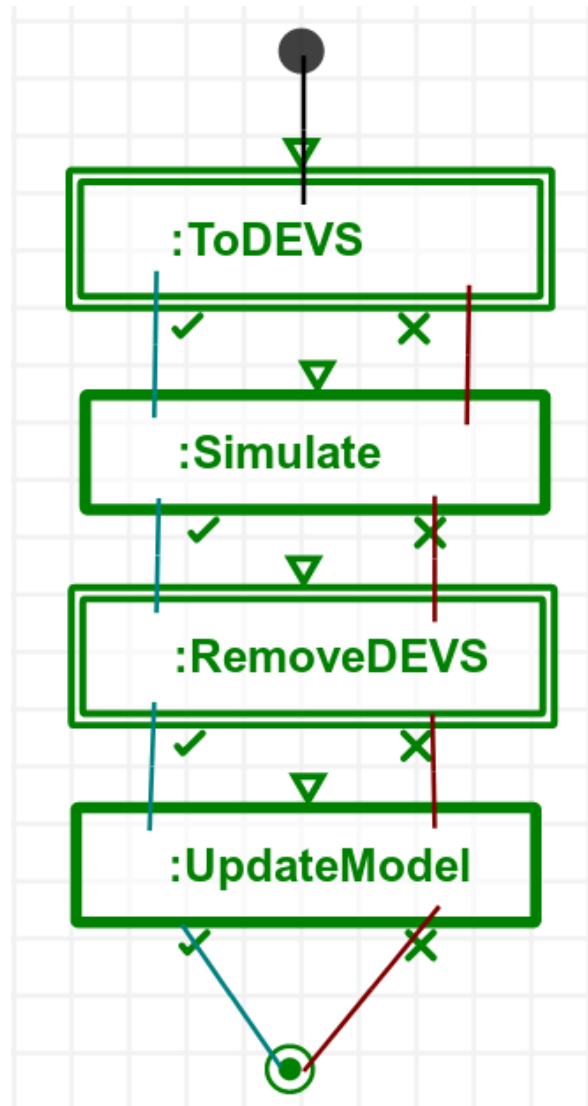
Properties

- Average transit time of schedule
- Throughput of track
- Average transit time of track

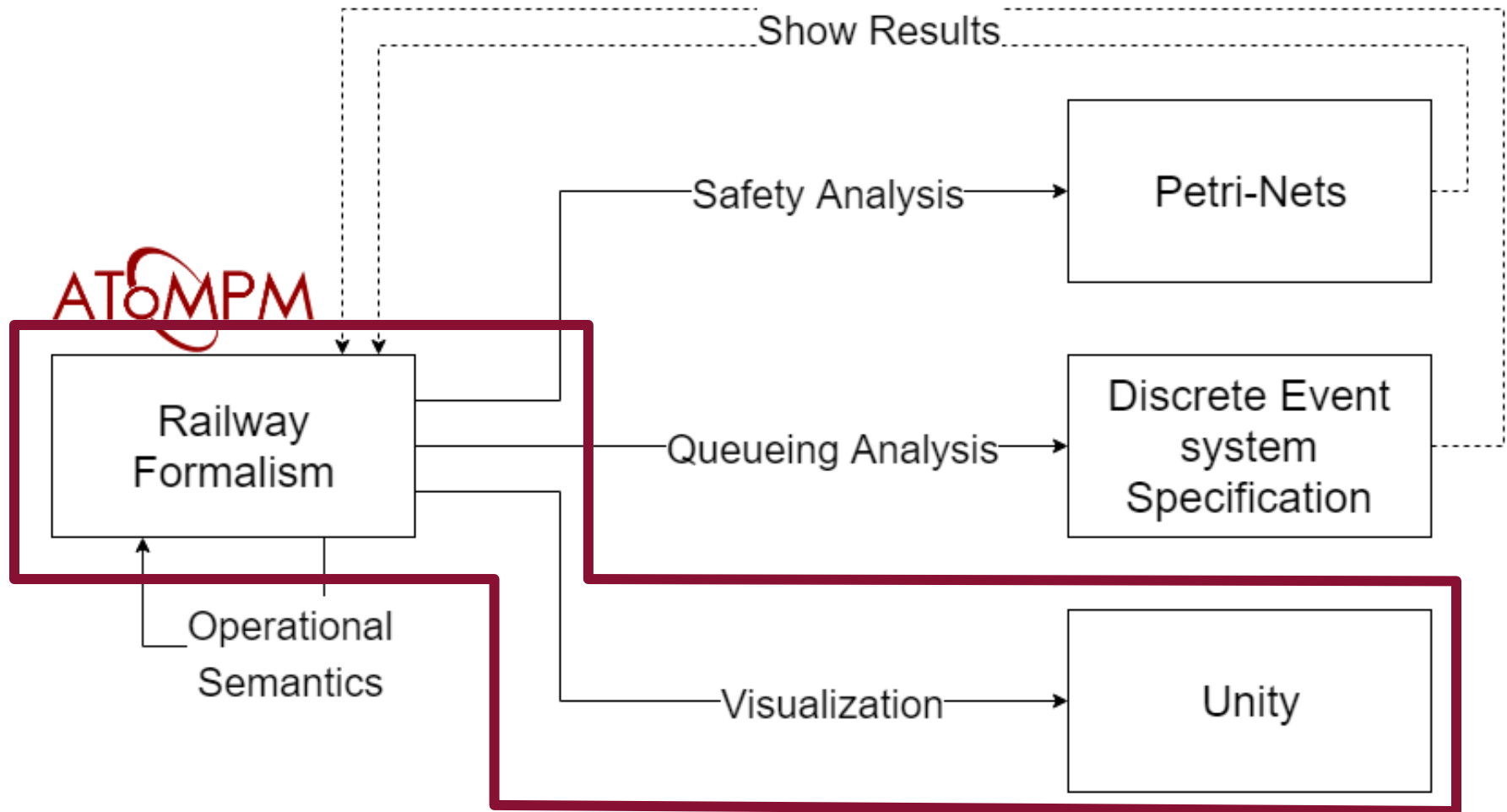
Interfaces

- Use ID's for traceability (\$atompmlId)
 - Generate python PythonPDEVS file
 - Call PythonPDEVS to run the simulation
 - Read results from file
- 
- Simulate
- UpdateModel

Schedule



5. Visualization



Model Generation

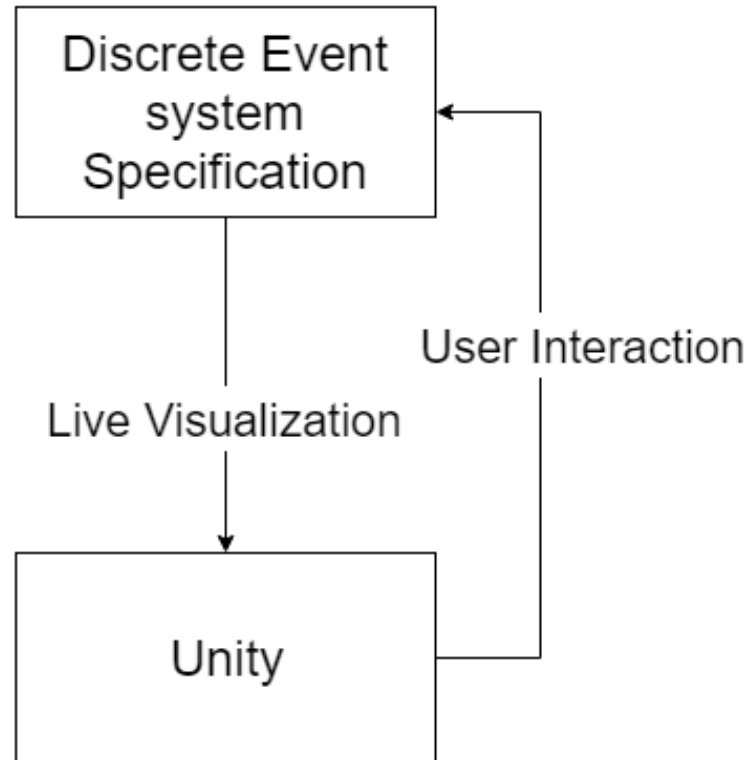
- Using Unity
- Small (xml) file to represent railway network
- Instantiate object in Unity

Simulation

- PythonPDEVS as simulator
- Custom tracer to create tracefile
- Read tracefile to resimulate model
- Gameloop:

```
currentTime = 0.0
while nextEventExists():
    if currentTime != nextTimestamp:
        wait (nextTimestamp - currentTime)
        currentTime = nextTimestamp
    simulateNextEvent()
```

6. Live Visualization and Interaction



Live Simulation

- PythonPDEVs as simulator
- Custom tracer to ~~create tracefile~~ send messages live to Unity (through sockets)
- Same messages as in tracefile:
 - No "gameloop" anymore

User interaction

- Tweak parameters during simulation
- Send message back to simulator from Unity
- Which model and what parameters to update
- Message interpreted as an external/user event in DEVS