Assignment 4
Translational Semantics (with traceability) in AToMPM

Joeri Exelmans
joeri.exelmans@uantwerpen.be

1 Practical Information

This assignment will teach you how to implement translational semantics in visual modelling tool **AToMPM**. You will apply this to the waterway network to allow for safety analysis with **Petri-Nets** (in the next assignment).

The different parts of this assignment:

1. Build a transformation schedule to generate a Petri-Net alongside the waterway network, connected by traceability links.
2. Build a transformation schedule that executes a Petri-Net transition, and updates the waterway network accordingly.
3. Create two (simple) waterway network models that are representative for all the features in your language. Show a few steps of the execution of these transformations for these models, and create a short video for one.
4. Write a report that includes a clear explanation of your complete solution and the modelling choices you made, as well as an explanation of your testing process. Also mention possible difficulties you encountered during the assignment, and how you solved them. Don't forget to mention all team members and their student IDs!

This assignment should be completed in groups of two if possible, otherwise individually is permissible.

Submit your assignment as a zip file (report in pdf + model files) on Blackboard before **14 December 2022, 23:59h**[1]. If you work in a group, only *one* person needs to submit the zip file, while all others *only* submit the report. Contact Joeri Exelmans if you experience any issues.

[1] Beware that BlackBoard's clock may differ slightly from yours.

2 Requirements

This section lists the requirements of the waterway network translation semantics. Make sure to test each requirement with test models!

2.1 Petri-Net Generation

Implement a rule-based transformation that generates a well-formed Petri-Net model from a waterway network (WN) model. The schedule that is part of your WN must become part of your PN (i.e., places and transitions). In other words, the Petri-Net that you generate must enforce the micro-step order that is defined in your WN schedule.

Use the 'PNS' (Petri-Net + Schedule) formalism (download the formalism here: http://msdl.uantwerpen.be/people/hv/teaching/MSBDesign/examples/PNS.zip from the assignment page). Note that the PNS formalism also allows capacity-constrained places. Do not generate instances of 'ScheduledTransition' (part of 'PNS') in this part of the assignment!

- Ensure that this transformation is complete. That is, a well-formed production system will produce a well-formed Petri-Net.
- Compared to the previous assignment, there are some simplifications to the semantics:
  - You do not have to limit the number of moves that a single watercraft can make in a macro-step to one.
  - Any segment type (source, segment, sink and confluence) may decide (non-deterministically) to skip its micro-step, even if a watercraft can be moved.
  - At a confluence, fairness is ignored: an input is chosen non-deterministically. At a confluence, watercraft are still only allowed to move "straight" (in other words, watercraft are not free to choose which output they take).
  - The *rate* of a source is only a restriction on how many watercraft it can produce over time. For instance, if a source has rate 2, it can produce no more than $n/2$ watercraft in n steps. In other words, the source's potential to produce watercraft accumulates whenever the source does not produce watercraft.
  - As always, ensure there can be at most one item on every segment or confluence. You are allowed to use capacity-constrained places from the 'PNS' formalism to enforce this.
  - Hint: First generate the Petri-Net places for all your segment types. Afterwards, generate Petri-Net transitions for all connection types. Finally, you can translate the WN schedule (that specifies the micro-step ordering) to your Petri-Net.

  - Hint: Maybe first create a Petri-Net for a WN by hand, in a tool like TAPAAL, to have an idea of what you want to arrive at, and then work out the rules.

- Note that inhibitor arcs cannot be used! This is due to the analysis tool in the next assignment.

- Do not remove your waterway network during this transformation! It is instead left untouched, co-existing with your Petri-Net. This allows a visual correspondence between both formalisms.

- Traceability links must be created from waterway network elements and to the Petri-Net *Places* and *Transitions*, such that the transformation in Section 2.2 can operate on both models **at the same time**.

  - Use the `/Formalisms/GenericGraph` formalism for traceability links.
  - You are allowed (and encouraged) to alter the concrete syntax of the `GenericGraph` formalism to show the arrowhead (all links in AToMPM are directed!), which will help you when creating your transformation rules. Don't forget to re-compile your metamodel!

- *Places* and *Transitions* must be uniquely named in the Petri-Net model.

  - Make sure the important *Places*/*Transitions* can easily refer back to their segments via their name. For instance, if you create a PN transition for a segment "`Seg1`" that represents movement of a watercraft onto the segment, you can call it "`EnterSeg1`".

- The counters of sources and sinks also have to be represented by Petri-Net places. Additionally, you have to implement a step counter, that counts the number of executed macro-steps. We will use these in the next assignment for analytical purposes.

- Layout considerations do not have to be considered in this transformation. That is, you may assume that the user will manually move Petri-Net elements to an appropriate location as they are created.

- **Warning:** There is a bug in AToMPM where the attributes of the associations are not reset to `result = True` or `result = getAttr()` when it is placed in the LHS, RHS, or NAC. You will have to **manually change the action code for all these created associations**. If you don't, rules will silently fail. Apologies for the inconvenience.

2.2 Executing the Petri-Net

You will now implement a transformation that executes simulaneously the Petri-Net, and updates the state of your waterway network to reflect the changes in the Petri-Net.

- Petri-Nets are non-deterministic. For instance, every segment can choose whether to execute its micro-step (if it can), or to skip it. Another example is the confluence: when a watercraft can enter from both inputs, one can be chosen arbitrarily. When there are multiple enabled transitions, we want the power to tell AToMPM which one to choose. Therefore, the 'PNS' formalism allows the modeler to specify a schedule of transitions to fire. See the example in `/Formalisms/PNS/samples/capConstraint.model`.
  - Do not confuse the Petri-Net schedule (a sequence of ScheduledTransition's) with your WN schedule. Your WN schedule is to be encoded in your generated Petri-Net, not as a PN schedule. It is only once you have your Petri-Net, that the PN schedule tells AToMPM which transitions to fire. See figure 1 for an overview.
- The 'PNS' formalism already includes a transformation that executes a modeled sequence of Petri-Net transitions. Every 'run' of the transformation executes one transition in the PN schedule, and then selects the next transition (for the next run). Your job is to embed this Petri-Net transformation into your own schedule (use the 'CRule' construct from MoTiF) and keep the waterway network 'in sync' with the Petri-Net.

3 Report

There are a number of requirements for the report. Above all, the reader must be able to read the report and have a clear understanding of your solution, without having to investigate the model files. I.e., your model files will only be used as a support for your report, not the other way around!

Specifically, the report must contain:

- A brief outline of your transformation rules and their schedules, and how they implement the requirements of the assignment.
- A discussion of any interesting decisions made.
- A discussion of possible improvements to the rules and transformation syntax.
- Two example models. For each, show:
  - Screenshots of at least a few steps of the waterway network during the generation of the Petri-Net, and its execution. If it helps to understand what is going on, highlight parts of your figures.
- Choose one waterway network and produce a short screen recording of the Petri-Net execution transformation running and showing interesting behaviour.

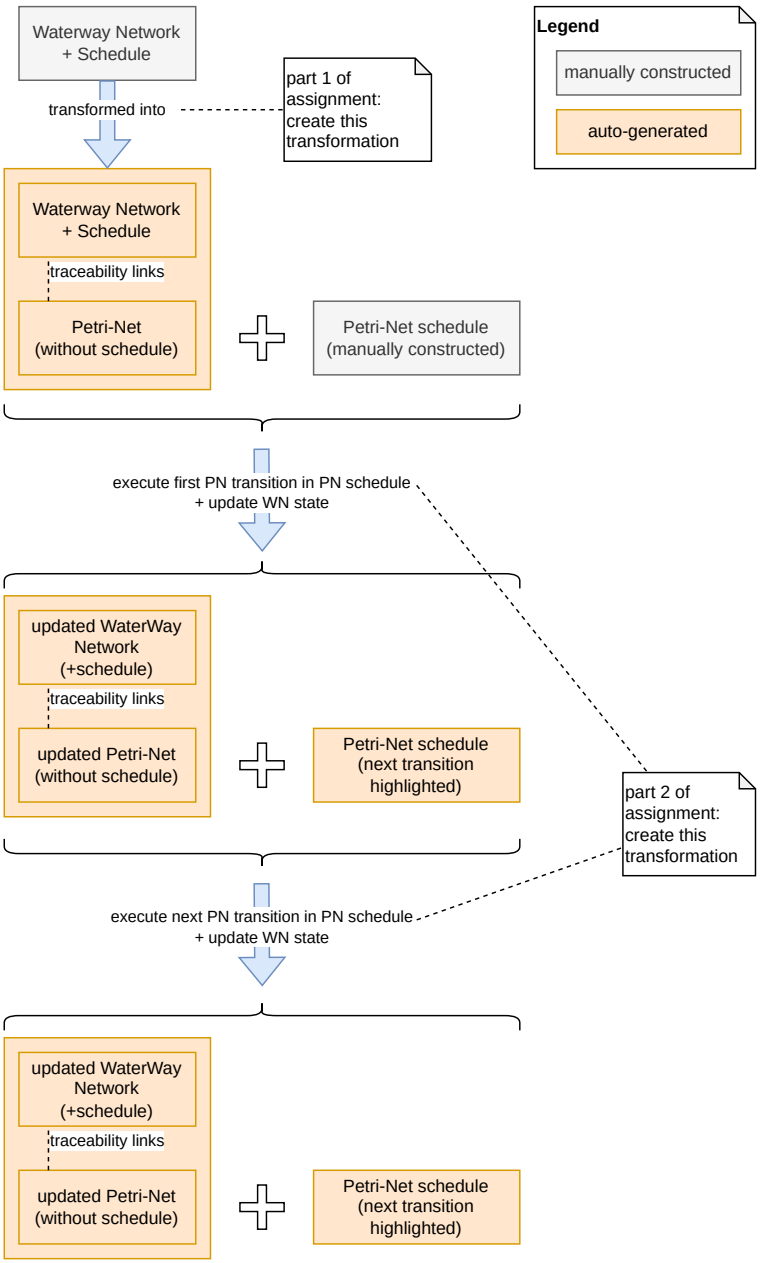

Figure 1: Overview of the different transformations that are part of this assignment.

- This video should not be submitted with your assignment (due to a large file size), but a link to where your video can be downloaded[2] should be placed in your report.
- For instance, you can upload it Dropbox, Google Drive, MS Azure, ...
- You can use OBS (https://obsproject.com/) or any other screen recording software.
- Example for last year's assignment: https://msdl.uantwerpen.be/cloud/public/d467f2

4 Useful Links and Tips

- AToMPM main page: https://atompm.github.io/
- Download and code: https://github.com/AToMPM/atompm
- Documentation: https://atompm.readthedocs.io/en/latest/
- This assignment will be used as an entry point for the next assignment. Therefore, it is important you finish this entire assignment.

Acknowledgements

Based on earlier assignments by Randy Paredis and Bentley Oakes.

[2] Must be downloadable for archival purposes.