Assignment 4
Translational Semantics in AToMPM

Randy Paredis
randy.paredis@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 production system to allow for safety analysis with **Petri-Nets**.

The different parts of this assignment:

1. Build a transformation to generate a Petri-Net alongside the production system model, connected by traceability links.
2. Build a transformation which executes the Petri-Net and visually shows the changes on your production system.
3. Create two (simple) production system 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 **7 December 2021, 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 Randy Paredis if you experience any issues.

2 Requirements

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

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

2.1 Petri-Net Generation

Implement a rule-based transformation that generates a well-formed Petri-Net model from a production system model.

- Ensure that this transformation is complete. That is, a well-formed production system will produce a well-formed Petri-Net.
  - As always, ensure there can be at most one item on every segment and in every machine.
  - Note that inhibitor arcs cannot be used! This is due to the analysis tool in the next assignment.
  - Hint: First generate the Petri-Net for the segments. Afterwards, you can use these components to generate the Petri-Net for the machines.
- Do not remove your production system during this transformation! It is instead left untouched, co-existing with your production system. This allows a visual correspondence between both formalisms.
- Traceability links must be created from the production system element and the created *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, which will help you when creating your models. 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 machines/segments via their name. For instance, a segment called "`Seg1`" has a place that identifies its contents, called "`P_Seg1_Items`".
- Simplifications:
  - Ignore any and all logic that has to do with the operators. Assume there will always be an operator at every machine, at all times.
  - You do not have to model the *Split* segment.
  - Assume all possibilities of the *Inspection* machine have equal weight. That is, do not model the probabilities.
  - Ignore *fairness*. You can assume a user will only move an item one segment every iteration of the simulation. Machines will also only be executed once in that time period. You do have to describe *how* a user would do this.

  - 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 action code on 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.

- Figure 1 shows a simple production system and the generated Petri-Net mid-simulation.

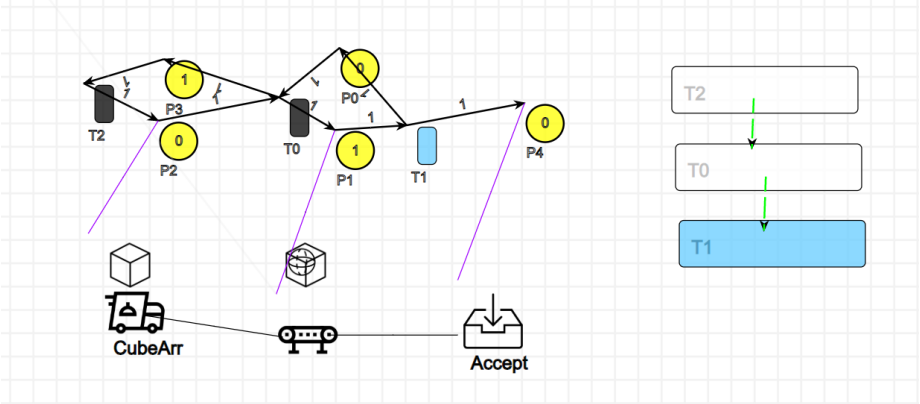

Figure 1: An example of a simple assembly line and the created Petri-Net.

2.2 Executing the Petri Net

Implement a rule-based transformation that executes the Petri-Net, and simultaneously visually updates the production system model.

- This basic Petri-Net execution transformation detects transitions, places a pivot on the transition to fire, consumes attached tokens, and then produces tokens.

  - Hint: You could add an "`active`" attribute to your transitions as a pivot. Provide a visual callback to mark this "active" transition.

- Make sure to add some rules that perform a visual update of the connected production system elements.

- Run-time info (items accepted, number of timesteps,...) must be represented in the Petri-Net by *Places*. These will be used in the following assignment for analysis.

- As the fireing of transitions in Petri-Nets is non-deterministic, you are required to extend the existing formalism, enforcing determinism. Implement a way to define a sequence of transitions to fire in an execution. You can assume they are modeled before the transformation is loaded. Base yourself on the `/Formalisms/PN` formalism. Feel free to extend it even more by adding positioning, snapping,... as needed. Don't forget to hand in your custom Petri-Net formalism as well.
- Simplifications:
  - To visually represent the state of the Petri-Net in the factory, it's enough to show *where* items are, not *which* item types they are.

3 Report

There are a number of requirements for the report. Above all, the marker must be able to read the report and have a clear understanding of all aspects of the assignment, 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 how the rules, transformations, and example models meet 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 production systems.
- For each production system, show:
  - Diagrams of at least a few steps of the production system during the generation and execution transformations. Highlight the items being assembled, destroyed, fixed, etc. and explain the behaviours you are showing.
  - These diagrams and your explanations must convince the reader that your transformation implements the translational semantics.
- Choose one production system and produce a short screen recording of the Petri-Net execution transformation running and showing interesting behaviour.
  - This video should not be submitted with your assignment (due to a large file size), but a link to where your video can be watched should be placed in your report. Note: this is not a download link!

  - For instance, you can upload it to YouTube, Vimeo, Google Drive, Dropbox,... Note that it should be unlisted, so it cannot be found, except when using the link.
  - A short description could be provided below the video, but no captions/voiceover/editing is required.
  - You can use OBS (https://obsproject.com/) or any other screen recording software.
  - Example: https://msdl.uantwerpen.be/cloud/public/d467f2
  - As in the second assignment, your solution should be more visually pleasing than the figure and video shown here. Please spend time introducing proper sizing, colouring, and placement of elements to make a clear visualization.

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 an earlier assignment by Bentley Oakes.

Icon authors from www.flaticon.com:

- Sphere - https://www.flaticon.com/authors/good-ware
- Cube, Receiver - https://www.flaticon.com/authors/smashicons
- Belts, Machine, Inspector, Incinerator - https://www.flaticon.com/authors/freepik
- Assembler - https://www.flaticon.com/authors/catalin-fertu
- Fixer - https://www.flaticon.com/authors/srip
- Walk - https://www.flaticon.com/authors/vitaly-gorbachev