Assignment 3
Production System Operational Semantics in
AToMPM
Bentley James Oakes

1 Practical Information

The goal of this assignment is to build a rule-based transformation for executing
the operational semantics of the production system modelling language in the
visual modelling tool AToMPM.

The different parts of this assignment:

1. Build the RAMified domain-specific language for the production system

2. Build the transformation rules. These rules must visually update the
production system model according to the operational semantics defined
below.
   • The formalism for this part will be
     Formalisms/Transformations/TransformationRule/TransformationRule

3. Schedule the transformation rules
   • The formalism for this part will be
     Formalisms/Transformations/Transformation/MoTiF

4. Create two production system models that are representative for all the
features in your language. Show a few steps of the transformation execu-
tion on these models, and create a short video (see below)

5. Write a report that includes a clear explanation of your complete solution
and the modelling choices you made.

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, commented syntax
and semantics models, and example models) on Blackboard before Tuesday,
November 10th, 23:59h. Contact Bentley Oakes (bentley.oakes@uantwerpen.be)
if you have any issues.
2 Requirements

This section lists the requirements of the production system operational semantics language and the report. Make sure to test each requirement with test models!

2.1 Abstract/Concrete Syntax Modification

Important: Feel free to modify your abstract syntax to be able to implement these operational semantics as transformations, or to make your solution more elegant. Your abstract syntax and operational semantics will not be compared against the earlier assignments.

The important part is that you clearly specify the choices you made. Explain why you changed the abstract syntax, if there another way but is too awkward/time-consuming, etc. This assignment is not about creating the perfect abstract syntax/operational semantics, but instead about being able to reason about (and discuss in the report) how information should be divided into these different syntax/semantics models.

In particular, sometimes it is difficult to represent operational semantics using rules. Your report should discuss the effects of the choices you made such as any non-determinism caused by rule matching.

2.2 Operational Semantics

This part of the assignment is copied from Assignment 1 for your convenience.

In this part of the assignment, the semantics of the production system will be modelled, including the Operators, Items, and Machines. The goal is for the Operators to move between Machines and operate them, such that Items are assembled, inspected, fixed, received, or destroyed.

The specific requirements are:

- The simulation is broken up into a number of discrete steps. In each step, the Machines are operated if Items and Operators are present, the Operators are moved if needed, and then the Items are moved (concurrently).
- In the initial step, all Operators are placed at their start Machine.
- If an Operator is scheduled to move to an occupied Machine, they must wait until the other Operator is finished.
- An Item is allowed to move to the next segment if no Item is present on that Segment, and the Item is not in an Assembler waiting for the linked Machine to obtain its Item.
- An Arrival cannot be operated if an Item is already on that Segment.
- In an Assembler when it operates, both Items are removed, and one is replaced with an AssembledItem.
At a Split, Cubes will take the straight direction, while Cylinders will take the diverging direction.

At a Join, one of the incoming Items is selected randomly to advance.

At an Inspector, the chance of the Item being accepted (placed on the accepted belt) is 70%. The chance of requiring fixing is 20%. The chance of the Item requiring destruction is 10%.

The simulation continues until some number of AssembledItems are accepted. Set this parameter such that the transformation execution is long enough to show interesting behaviour.

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.

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 as it is transformed. 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 operational semantics.
  – Note that (textual) traces are not required for this assignment, as the visual representation of the production system should show the desired behaviour.
• Choose one production system and produce a short screen recording of the transformation running and showing interesting behaviour.
  – This video should be uploaded to YouTube and the link placed in the report. Note that it should be unlisted, so it cannot be found except for the link.
  – A short description should be provided below the video, but no captions/voiceover/editing is required.
  – Example: https://youtu.be/QenXgaqKOSg
• 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/
• For moving elements, there is an example rule in Formalisms/RaceCar.
• Consider using a third run-time language for storing run-time information
• If you change the abstract or concrete syntax for a language, you may need to delete and recreate all elements that you changed. Thus it is strongly recommended that you work out the rules and languages on paper first, before creating rules in AToMPM.
• If a rule is failing:
  – Double-check the labels in the LHS, RHS, and actions
– Labels in the action must be strings. For example, `setAttr('position', [0, 0], '3')`. Note that this last argument is a string, not an integer!

– If problems persist, try deleting the element and recreating it.

Acknowledgements

Based on an earlier assignment by Simon Van Mierlo.

Icon authors from www.flaticon.com:

- Cylinder - https://www.flaticon.com/authors/kiranshastry
- Cube - https://www.flaticon.com/authors/smashicons
- Belts, Machine, Inspector, Incinerator, Receiver - https://www.flaticon.com/authors/freepik
- Arrival Machine - https://www.flaticon.com/authors/catalin-fertu
- Fixer - https://www.flaticon.com/authors/srip