

# Assignment 3

## Operational Semantics in AToMPM

Joeri Exelmans  
joeri.exelmans@uantwerpen.be

### 1 Practical Information

The goal of this assignment is to build a rule-based transformation for executing the operational semantics of the waterway network modeling language in the visual modeling tool **AToMPM**.

The different parts of this assignment:

1. RAMify your waterway network DSL, resulting in a new pattern language for your DSL.
2. Use the *compile an abstract syntax model (...) into a pattern metamodel* button.
3. Build the transformation rules. These rules must update the waterway network model according to the operational semantics defined below.
  - Create a new rule with the *create new rule* button. Then load your pattern language using the *load a pattern metamodel* button. The created rule will then be an instance of two formalisms: (1) `/Formalisms/_Transformations_/TransformationRule/TransformationRule` and (2) your pattern language.
4. Create a schedule for your transformation rules, such that executing the schedule on an instance of your waterway network DSL executes one step (altering the model in-place). The schedule is basically a program in a procedural language, that defines in what order to execute your rules, to choose alternatives depending on some condition, and possibly even executing certain sequences of rules in a loop, until some condition is met.
  - Create a schedule with the *create new transformation* button.
5. Create two waterway network models that are representative for all the features in your language. Show a few steps of the transformation execution on these models, and create a short video (see below).

6. Write a report that includes a clear explanation of your complete solution and the modeling 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. The deadline will be posted on the assignment web page <sup>1</sup>. If you work in a group, only *one* person needs to submit the zip file, while the other person *only* submits the report. Contact Joeri Exelmans if you experience any issues.

## 2 Requirements

This section lists the requirements of the waterway network operational semantics. Use test models to test the requirements. Ideally every test model only tests one requirement.

### 2.1 Abstract/Concrete Syntax Modification

**Important:** *You will probably have to modify/extend your abstract syntax (and concrete syntax) to be able to implement these operational semantics as transformations. In the MetaDepth assignment, we used run-time variables (in EOL code) to keep track of the runtime state (during the execution of a step). For instance, every waterway element was micro-stepped only once, and we had to keep track of which elements have already been stepped. In AToMPM, all of this runtime information needs to be an explicit part of your model.*

Make sure 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) rule-based model transformations.

### 2.2 Operational Semantics

In this part of the assignment, the semantics of the waterway network will be modeled, as a set of transformation rules, composed into a transformation schedule. Your schedule, when executed, should perform one macro-step on an instance of your DSL.

**Simplification wrt. Assignment 1:** You may now assume that Sources, Sinks and Confluences will only connect to ordinary Segments. In other words, it is not possible to connect a Source directly to a Confluence or Sink. (Motivation:

---

<sup>1</sup><http://msdl.uantwerpen.be/people/hv/teaching/MSBDesign>

If we allowed this, it would be too difficult (but still possible!) to express the operational semantics in a rule-based manner.)

The specific requirements are:

- During a macro-step, every watercraft is allowed to make at most one move.
- During a macro-step, every segment (i.e., ordinary Segment, Sink, Source, Confluence) performs one micro-step.
- The order in which the different segments in your network perform micro-steps, is modeled explicitly. This order is a part of your model and remains the same for all steps.

TIP: One way to define this order, is to put all your waterway segments in a linked list.

TIP: During the execution of a step, you will have to keep track of which segment's "turn" it is to execute a micro-step...

- Depending on the type of segment, a micro-step looks as follows:

**Sink** . If the Sink's input segment has a Watercraft *available* (what this availability means, is explained later), it consumes the Watercraft, removing it from the input segment, and incrementing the Sink-counter by 1.

**(ordinary) Segment** . If the Segment's input has a Watercraft available, and this Segment has no Watercraft, then this Watercraft is moved onto this Segment.

**Confluence** .

- If the Confluence already has a Watercraft on it, nothing happens (not enough space).
- Otherwise:
  - \* If only one input  $i_x$  of the two inputs of the Confluence has a watercraft on it, then that watercraft is moved onto the confluence, and the confluence's mode is set to  $x$ .
  - \* If both inputs have a watercraft on them, and the mode of the Confluence is  $x$ , then the watercraft from input  $i_{1-x}$  is moved onto the Confluence, and the mode is set to  $1 - x$ .  
This way, we implement *fair scheduling*, meaning that any input will be served eventually. One input cannot *starve* the other.
  - \* If none of the Confluence's inputs have a watercraft available, nothing happens.

**Source** . Does nothing.

- Depending on the type of segment  $A$ , it has a *Watercraft available* for another segment  $B$  if:

**Source** . True every  $r$  steps, otherwise false, where  $r$  is the Source's rate. For instance, if the rate is 1, then every step, a Watercraft is produced. If the rate is 2, then every other step.

Note: If during a step, a Source has a Watercraft available for the next segment, this does not mean that a Watercraft is actually produced. A Watercraft is only produced if the next segment is also able to take the Watercraft.

**(ordinary) Segment** . True iff the segment  $A$  has a Watercraft on it.

**Confluence** . True iff the Confluence  $A$  has a Watercraft on it, and  $B$  is output  $out_x$  of  $A$ , where  $x$  is  $A$ 's current mode.

**Sink** . Never.

- Recap (this is not new information): A Confluence's *mode* serves two purposes within the same step: First, when its outputs are being micro-stepped, the mode determines the output Watercraft should use to leave the Confluence, Next, (still in the same step) when the Confluence is being micro-stepped, it determines which input has the *lowest* priority.

Think carefully about what information should be stored in the abstract syntax of the model, such as information that is relevant to the simulator. You are free to choose how much information is stored in your model, as long as all information is useful in some way.

### 3 Report

There are a number of requirements for the report. Above all, it must be able to read the report and have a clear understanding of all aspects 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 how the rules, transformations, and example models meet the requirements of the assignment.
  - This may include meta-models, diagrams, (pseudo-)code, etc. as needed to provide the essential details of the assignment.
- A discussion of any interesting decisions made.
- A discussion of possible improvements to the rules and transformation syntax.
- Two example waterway networks. For each of them, show:
  - One figure of the model *before* executing a step.
  - At least one figure of the model during step execution. (Explain what is going on!)

- One figure of the model after having executed the step.
- Choose one waterway network and produce a short screen recording of the transformation running and showing interesting behaviour.
  - This video should not be submitted to Blackboard (too large file size). Instead, attach a link to some hosting platform (e.g., Dropbox, Google Drive, ...) where your video can be downloaded.
  - You can use OBS (<https://obsproject.com/>) or any other screen recording software.
  - Example: <https://msdl.uantwerpen.be/cloud/public/5cbdeb> (note: video is from previous year's assignment)

## 4 Useful Links and Tips

- AToMPM main page: <https://atomp.m.github.io/>
- Download and code: <https://github.com/AToMPM/atomp.m>
- Documentation: <https://atomp.m.readthedocs.io/en/latest/>
- (For moving elements, there is an example rule in Formalisms/RaceCar.)
- 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 unexpectedly 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 the NAC is empty, delete the NAC-node. An empty pattern will always match, and as a result an empty NAC will always cause the rule to fail.
  - If problems persist, try deleting the element and recreating it.
- More tips (at the bottom): <http://msdl.uantwerpen.be/people/hv/teaching/MSBDesign/scribbles/AToMPM-summary-and-tips.txt>

### Acknowledgements

Based on an earlier assignments by Randy Paredis and Bentley Oakes.