TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Software and Multimedia Technology, Software Technology Group

WS2020/21 – Model-driven Software Development in Technical Spaces

Model-to-Model Transformation

Professor: Prof. Dr. Uwe Aßmann
Tutor: Markus Hamann

1 Model-to-Model Transformation with Epsilon

This exercise provides a tutorial on model-to-model transformations. These transformations, as the name suggests, transform models to other models of the same or different metamodels. In this exercise, we will focus on model-to-model transformations using the *Epsilon*[1] language family [1], particularly the *Epsilon Transformation Language* (ETL) [2], to transform our *Statechart* models into *Petri Net* models.

1.1 Petri Net Metamodel

To transfer our statechart models to petri net models we need a *Petri Net Metamodel*. The first task of this exercise is to define an EMF metamodel for simple *Elementary Perti Nets*[2] like it was done in Exercise 1.

Our petri nets are defined as the following:

Each *Petri Net* has a *name* to identify it. It has multiple *Places*, which each place can have up to one *Token*. Another element of the petri net is the *Transition*, symbolizing passages between places. Both places and transitions are defined by their *id*. Places and transitions are connected via directed *Arcs*. There are *Incoming Arcs*, connecting a place with a transition, and *Outgoing Arcs* connecting a Transition with a Place.

[1] https://www.eclipse.org/epsilon/
[2] http://st.inf.tu-dresden.de/files/teaching/ws18/swt2/slides/02-st2-colored-petri-nets.pdf

1.2 Epsilon Transformation Language

Epsilon is a language family that can be used for many different types of model processing. The *Epsilon Generation Language* (EGL), for example, is a model-to-text alternative for *Acceleo* (see Exercise 4). In this exercise, we will focus on the *Epsilon Transformation Language* (ETL), which we will use to transform our *Statechart* models into simple *Petri Net* models.
At first, you will prepare an Epsilon ETL environment:

- Install *Epsilon 2.2* from the *Eclipse Marketplace*
- Open a eclipse workspace with your statechart metamodel projects, and petri net metamodel projects (Metamodel, Edit, and Editor) or import them in a new workspace. *Run* the Metamodel Project *as* an *Eclipse Application* to register both metamodels in Eclipse. In the new Eclipse instance you can start using Epsilon.
- Epsilon does not need specific projects to work. Create a new *general project* in your workspace and create a new `*.etl` file in it.
- You can now declare your model-to-model transformation rules in this `*.etl` file. To familiarize yourself with ETL, you can use this documentation[3][4] and examples[5].
- To run the transformation, *right-click* on the `*.etl` file and choose *Run As -> Run Configuration*. In the next window create a new *ETL Transformation* configuration, select your `*.etl` file under *Source*, and add both your *source* statechart model and your *target* petri net model (for the beginning just create an empty one) under *Models* (as EMF Models) (Hint: deselect *Read on Load* for the target model). Now you can always *Run* the configuration and start the transformation.

1.3 Statechart-to-PetriNet Transformation

At last, declare the transformation rules in the `*.etl` file to transform your statechart models to petri net models. Test your ETL transformation on several statechart models from previous exercises.
Please note the following hints:

- At the start, you can ignore the *guard*, and *actions* of *Transitions* as well as *Activities*.
- If this transformation is working, try to add transformation rules for the *guard* and *action* of Transitions.

If you wish a review of your solution, please hand in the `*.etl` file and your *Petri Net metamodel* till the day before the next exercise (E-Mail: `Markus.Hamann1@tu-dresden.de`). If you did not hand in your metamodel after Exercise 1, you need to hand it in too. As always, on the day of the next exercise, an example solution will be published.

[3] https://www.eclipse.org/epsilon/doc/etl/
[4] https://www.eclipse.org/epsilon/doc/eol/
[5] https://www.eclipse.org/epsilon/examples/#epsilon-transformation-language

1.4 (Optional) A Model Transformator Based on Your Project

Like in the last exercise, we want to encourage you to create your own model-driven toolchain. If you created a metamodel from one of your projects, try to create a Transformator to other model types (for example UML) for it.

References

[1] Dimitrios Kolovos, Louis Rose, Richard Paige, and A Garcıa-Domınguez. The epsilon book. *Structure*, 178:1–10, 2010.

[2] Dimitrios S Kolovos, Richard F Paige, and Fiona AC Polack. The epsilon transformation language. *ICMT*, 8:46–60, 2008.