A Domain-Specific Language for Lego Vehicles

Mark van den Brand, Luc Engelen, and Gerard Zwaan

October 8, 2013
1 Introduction

The goal of the assignments described in this document is developing a small domain-specific language (DSL) to control Lego vehicles and make them display their path on their screens. The DSL that you will develop describes named lists of instructions for these vehicles. Each instruction instructs a vehicle to move a certain distance in a given cardinal direction. For instance, the instruction move \((N, 5)\) instructs a vehicle to move 5 units to the north. To achieve the goal of controlling the vehicle with this language, you are asked to develop a number of (auxiliary) DSLs, model transformations, and a template for code generation.

For this assignment, you will work with Eclipse and the Eclipse extensions Xtext, the Eclipse Modeling Framework (EMF), and the Epsilon toolkit. Installation instructions for Eclipse and its extensions are provided in a separate document.

2 First Assignment

The first assignment is meant to get you acquainted with the basics of EMF. You are asked to construct two Ecore models that serve as metamodels for two small languages. In a later assignment, you will define the textual syntax for these languages.

To complete this assignment, create two Ecore models that correspond to the diagrams below, following the tutorial on metamodeling that is provided as a separate document. Create an Ecore model named Coordinates.ecore that represents the metamodel of the Coordinates language, and an Ecore model named News.ecore that represents the metamodel of the News language.

![Figure 1: Metamodel for the Coordinates language](coordinates_metamodel.png)

![Figure 2: Metamodel for the News language](news_metamodel.png)
3 Grammars and Metamodels Workspace

In an archive that is distributed with this document, you can find two workspaces for Eclipse. Unpack this archive and open the grammars-and-metamodels workspace in Eclipse before you continue with the next assignment.

The workspace contains a number of plug-in projects. The plug-in project that is named nl.tue.glt1314.coordinates contains a fragment of the grammar for the Coordinates language, the project nl.tue.glt1314.news contains a fragment of the grammar for the News language, and the project nl.tue.glt1314.movements contains the metamodel of the Movements language.

4 Second Assignment

For the second assignment, you must complete the grammars of the Coordinates language and the News language. You can use the grammars-and-metamodels workspace to do this.

The following model (or program) is written in the News language:

Box:
move( N, 5 )
move( E, 5 )
move( S, 5 )
move( W, 5 )

The following model (or program) is written in the Coordinates language:

Random:
(0, 0)
(-4, 5)
(88, -5)

The grammars in this workspace are defined using Xtext. A manual for Xtext and a number of tutorials are available on-line.

After completing a grammar, you generate code from this grammar by executing a workflow that is located in the same folder as your grammar. Each time you execute this workflow, a metamodel based on your grammar is generated as well. You can find this metamodel in the model folder of the plug-in of your grammar.

To complete this assignment, complete the grammars and ensure that the corresponding metamodels are the same as those created in the previous assignment. For the following assignments you also need to export the projects as plugins.

Hand in the results for the 1st and 2nd assignment before Friday October 18th 23:45h. Material to be handed in should be a zip-file containing the workspace with the.ecore models and the adapted workspace grammars-and-metamodels with the filled in Xtext grammars.

5 Models and Transformations Workspace

In the models-and-transformations workspace from the archive distributed with this document, you will find a number of models, (fragments of) model transformations, and (fragments of)
templates for code generation. The transformations are implemented in ETL and the templates in EGL, both of which are part of the Epsilon toolkit\(^3\). A manual, examples, and tutorials for all the languages in the Epsilon toolkit are available on the Epsilon website.

You can execute each of the transformations and templates by running the corresponding Ant scripts in the script folders of the plug-in projects. The scripts contain a description on how to run them.

6 Third Assignment

The third assignment requires you to complete the two model transformations \texttt{news2coordinates} and \texttt{news2movements}.

The \texttt{news2coordinates} transformation takes a model written in the \texttt{News} language and transforms it to a model written in the \texttt{Coordinates} language. The result is a named list of coordinates that contains every start and end point of the instructions in the \texttt{News} model. You can assume that the starting point is located at coordinate \((0, 0)\) and the vehicle faces north.

The \texttt{news2movements} transformation takes a model written in the \texttt{News} language and transforms it to a model that conforms to the \texttt{Movements} metamodel. A \texttt{Movements} model is a named list of movements. A movement is either a turn movement or a forward movement. A turn movement specifies that the vehicle should make a turn with an angle of a given size specified in radians, and a forward movement specifies that the vehicle should move forward for a given amount of time.

In the desired-results folder of the models project, you can find a number of example models, which you can use to verify your transformations.

7 Lego Vehicle

The small Lego vehicle that is used for these assignments is described on pages 5 to 15 of the Lego Mindstorms NXT manual\(^4\). It should be programmed using NXC\(^5\). The Bricx Command Center\(^6\) can be used to compile NXC programs and transfer them to the NXT brick.

\(^3\)http://www.eclipse.org/epsilon/
\(^5\)http://bricxcc.sourceforge.net/nbc/nxcdoc/
\(^6\)http://bricxcc.sourceforge.net/
8 Fourth Assignment

The fourth assignment requires you to implement the EGL template `coordinates2NXC`, which generates NXC code from a `Coordinates` model. The generated code is supposed to display the path through a list of coordinates on the display of an NXT controller. The path must be properly scaled to fit the display and enclosed in a square frame that is as large as possible and centered on the display.

The `movements2NXC` template is provided as an example of an EGL template.

Hand in the results for the 3rd and 4th assignment before Friday October 25th 23:45h. Material to be handed in should be a zip-file containing the adapted workspace `models-and-transformations` with the filled in model transformations and generator template.