Assignment 3

Generic Language Theory

By Marc Craenen and Huib Donkers

Import project into Eclipse

In order to make sure the mm.zip file isn't too large for peach, we deleted some auto-generated files from the zip-file. In order to make everything work, you can follow the following steps:

- Copy the contents of `mm.zip` into a new empty folder
- Open Eclipse
- Switch workspace to the newly created folder
- Import → General → Existing projects into Workspace
- Select the newly created folder and select all projects
- Go to `assignment3b_boundingbox/src/nl/tue/glt/` and run `GenerateBoundingBox.mwe2`
- Do the same for `GeneratePlatoon.mwe2` in `assignment3b_platoon/src/nl/tue/glt/`

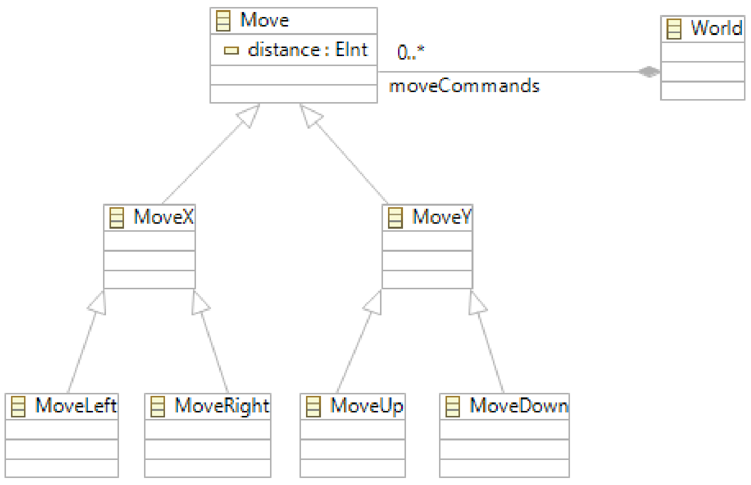
  Please note that Eclipse will give some errors while validating the `.xtext` files, because of the deleted files in `mm.zip`. This doesn't matter, since the `.mwe2` will still run correctly. It is just the Eclipse validation that goes wrong.

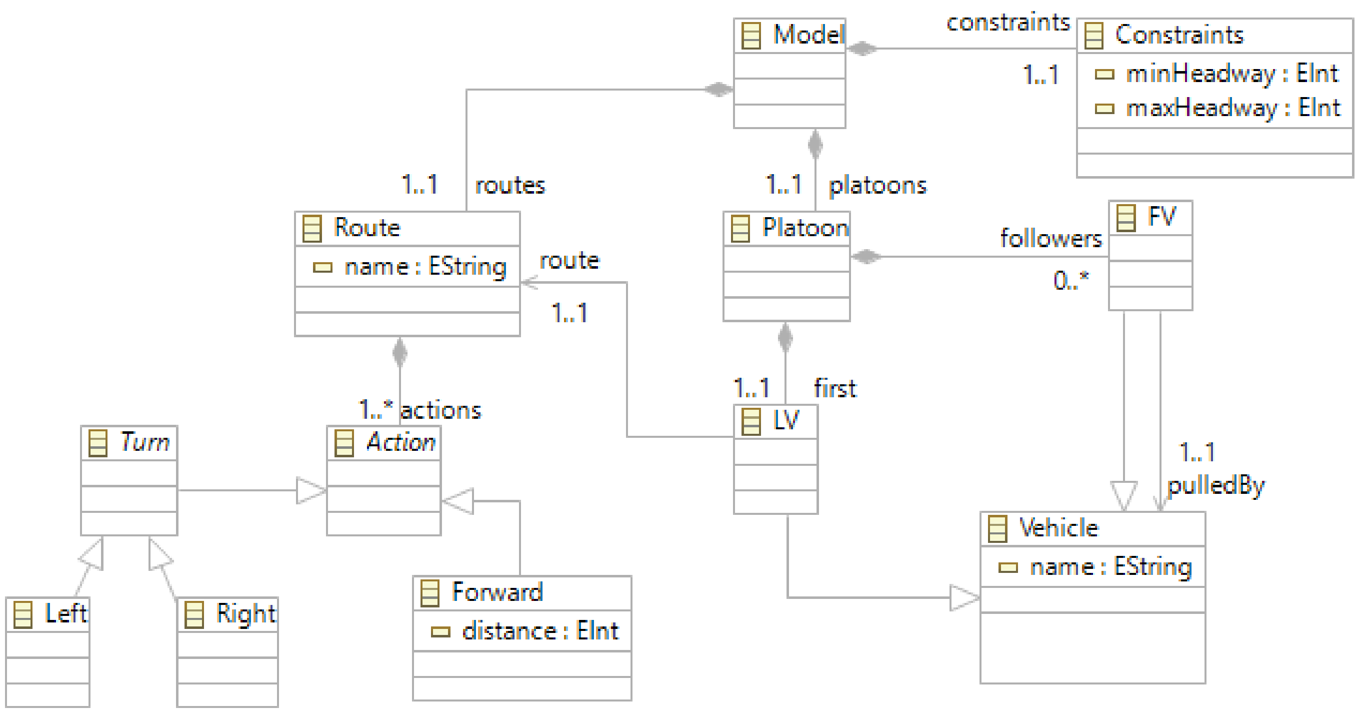
- Go to `assignment3b_platoon.ui/plugin.xml` and press "Launch an Eclipse application"
- Create a new empty project containing two files: `test.boundingdsl` and `test.platoondsl`, where you can write in the newly created languages in order to test the `.xtext` files. The editor probably will ask if you want to turn the project into an xtext-project. Agree with this.

First assignment

The metamodels for this assignment are found in the Eclipse project `assignment3` and are stored in the `/model` folder. The images below show the created metamodels for both languages. The decisions we made while creating the metamodel for the platoon language, are mentioned below as well.

- Since there is only one platoon, it makes sense to also have exactly one route and one constraint in the metamodel.
- Both LV and FV are vehicles that can pull other vehicles, but only FV can be pulled by one of these vehicles. Each FV can only drive behind exactly one vehicle.
- Each platoon should contain exactly one LV, but an arbitrary amount of FV.
- Each FV is only aware of its front runner. Its front runner doesn't have to be aware of FV.
- The LV follows only one route.
- Each route contains out of one or more actions, which can be either a turn (left or right) or a forward move over a certain distance.





Second assignment

In the `assignment3b_boundingbox` and `assignment3b_platoon` projects you can find the Xtext files that are used to describe the boundingbox and platoon language respectively. These projects are automatically generated by Eclipse, based on the created Ecore models and were edited to fit the description of the assignment.

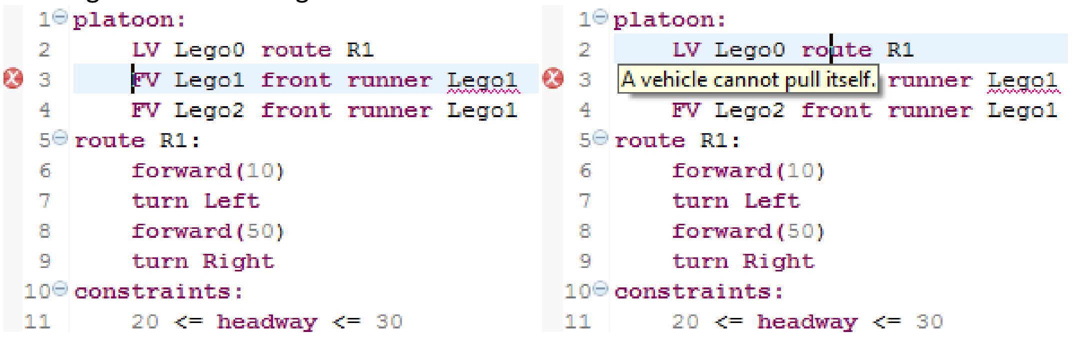
Most of the decisions were already made during the first part of the assignment. In order to prevent weird platoons, we added tree extra constraints in:
`assignment3b_platoon/nl/tue/glt/validation/PlatoonValidator.xtend`

Constraint 1

Obviously a vehicle can't be the front runner of itself, so we added the following code:

```
@Check
def checkNotPullingItself(FV fv) {
    if (fv.name.equals(fv.pulledBy.name)) {
        error("A vehicle cannot pull itself.", PlatoonPackage.Literals.FV__PULLED_BY);
    }
}
```

Which gives the following result:



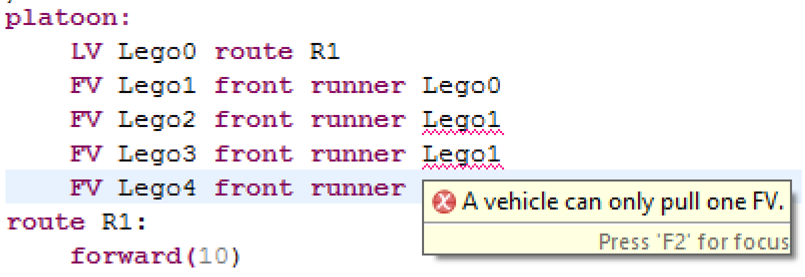
```
platoon:
    LV Lego0 route R1
    FV Lego1 front runner Lego1
    FV Lego2 front runner Lego1
route R1:
    forward(10)
    turn Left
    forward(50)
    turn Right
constraints:
    20 <= headway <= 30
```

```
platoon:
    LV Lego0 route R1
 A vehicle cannot pull itself. runner Lego1
    FV Lego2 front runner Lego1
route R1:
    forward(10)
    turn Left
    forward(50)
    turn Right
constraints:
    20 <= headway <= 30
```

Constraint 2

One vehicle should only pull one other vehicle, i.e. no two vehicles have the same front runners:

```
@Check
def checkSameFrontRunners(FV fv) {
    for(EObject obj : fv.eContainer.eContents) {
        if (obj != fv) {
            if (obj instanceof FV) {
                if (obj.pulledBy == fv.pulledBy) {
                    error ("A vehicle can only pull one FV.", PlatoonPackage.Literals.FV__PULLED_BY);
                }
            }
        }
    }
}
```



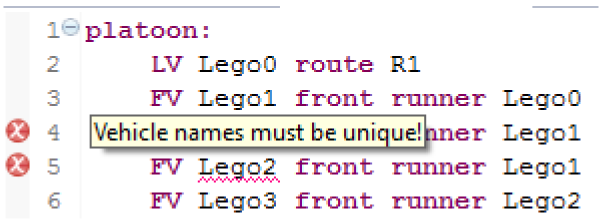
```
platoon:
    LV Lego0 route R1
    FV Lego1 front runner Lego0
    FV Lego2 front runner Lego1
    FV Lego3 front runner Lego1
    FV Lego4 front runner
route R1:
    forward(10)
```

A vehicle can only pull one FV.
Press 'F2' for focus

Constraint 3

In order to avoid confusion, we also each vehicle to have a unique name:

```
@Check
def checkUniqueName(Vehicle vehicle) {
    for(EObject obj : vehicle.eContainer.eContents) {
        if (obj != vehicle) {
            if (obj instanceof Vehicle) {
                if (obj.name == vehicle.name) {
                    error ("Vehicle names must be unique.", PlatoonPackage.Literals.VEHICLE__NAME);
                }
            }
        }
    }
}
```



```
platoon:
     LV Lego0 route R1
     FV Lego1 front runner Lego0
Vehicle names must be unique!nner Lego1
     FV Lego2 front runner Lego1
     FV Lego3 front runner Lego2
```