UNIVERSITÉ CÔTE D'AZUR
L3IA
2021-2022

Julien Deantoni

Language et Compilation

IziBot

Project delivery

This document concerns the project delivery of the Language and compilation course. It will include the implementation of a DSL focused on the specification and tooling of a language to program Lego mindstorm robots. Deliveries are expected by email (to Julien Deantoni: firstname.lastname@univ-cotedazur.fr, with [L3IA] as object prefix) followed by "firstname lastname" (where firstname and lastname are your actual names !). The delivery is expected before the 7^{th} of March 2022 at 10:00PM Paris Time. The delivery is expected as a PDF report (please let your report be succinct and rigorous).
The report must contain :

- a link to the code of your language (typically a link to the git repo)
- a description of the language proposed:
  - the domain model represented as a class diagram;
  - the concrete syntax represented in a BNF-like form;
  - a description of your language and how it was implemented;
  - a simple description on how you wrote the compiler to obtain executable code
- a set of relevant scenarios implemented by using your language;
- a rationalization of its main usage and all of its features. The associated script and materials will be provided.

Objectives: Define the IziBot language

Your objective here is to define a tooled language allowing the definition of various, possibly complex control of Lego Mindstorm robots. The provided language should allow an easy and intuitive control of the robots as well as the definition of how the sensors and actuators are actually connected to the robot (see for instance here for minimal information: `https://slidetodoc.com/presentation_image_h2/492008e00e08fe2302fd63423955e79d/image-7.jpg`).
Basic scenarios are defined below and must be supported by your language. However, you should define more complex scenarios and keep them in your repository. You should at least provide one program to illustrate each functionality of your language. Keep in mind that the goal is to provide an easy and intuitive language to program the robots.

The programs written in your language should in fine generate code that complies the `ev3dev2` python library[1]. In order to test your code, you can use the Gears simulator accessible here: `https://gears.aposteriori.com.sg/`.

Basic scenarios

1. **Line Following Example**

   Alice wants to use two color sensors to define a control where the robot follows a line as defined at the beginning of this video : `https://youtu.be/14VhPmGtDWE`. She wants to put the left motor on output *A*, right motor on output *B*. The two colors sensors are plugged in input 1 and 2. Depending on the value of the sensors, she will put more power on the left or the right motor.

   Remark: This robot is named "Double Sensor Line Follower" in the online gear simulator. A world with a line can also be chosen.

2. **Paintball Challenges**

   Bob wants to reuse the Robot from Alice but he will add 2 sensors. First he will plug the gyro sensor in input 4. He will also plug a Laser Range sensor in input 3. He will also add two more actuators: one motor to control a paintball gun angle and one motor to actually fire the paint ball. The motors will be respectively plugged in outputs *D* and *E*. He does to actually realize the paintball challenges as introduced in this video: `https://youtu.be/bwABggKJDEE`

Commmon Parts

The following parts must be available in your DSL:

- **Domain Model**:

  The domain model (a.k.a. abstract syntax) should be clearly identified in the delivered code. It will be provided as a class diagram, together with explanation about the main choices you did to propose easy and intuitive construction to control a lego robot.

- **Concrete syntax**:

  The concrete syntax, in a eBNF form, must be clearly identified and used by a relevant set of scenarios. The syntax must introduce the textual constructions you proposed in your language in order to make it easy and intuitive.

- **Validation**:

  Support your end-user by checking that a model is realizable on the expected platform. For instance you cannot have more then *n* actuators or *m* sensors plugged in at the same time.

- **Code generation**:

  Provide a generator producing code that can be copy and paste in the online simulator.

Any of the points above may be discussed between you and me during the course hours or on the slack.

[1] `https://github.com/ev3dev/ev3dev-lang-python`