Alchemist Simulation Basics

Giorgio Audrito

January 28, 2020

1 Setup

1. Install the virtual machine *Java SE 11 JDK*:

   `oracle.com/technetwork/java/javase/downloads`

2. Install the *Eclipse* or *IntelliJ* IDE:

   `eclipse.org/downloads`
   `jetbrains.com/idea/download`

   Eclipse has better support for Protelis (buggy on ScaFi), while IntelliJ has better support for ScaFi (no Protelis syntax highlighting).

3. Install a YAML support plugin of your choice.

4. If you plan to use Protelis on Eclipse, install the *Protelis* plugin from `Help > Eclipse Marketplace` for syntax highlighting (not available for IntelliJ).

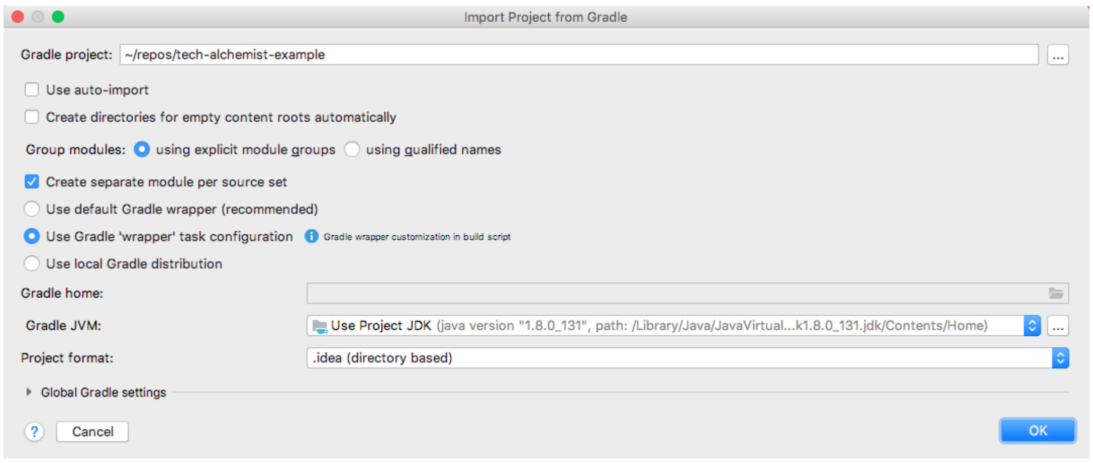
5. If you plan to use ScaFi on Eclipse, install the *Scala IDE* available from `Help > Eclipse Marketplace`. For using ScaFi on IntelliJ, click on `File > Project Structure > Global Libraries`. In the window that will open, search for `Create > Download`, then select version `2.12.3` and press `OK`. Wait for the download to end then press `OK` again.

6. Download the sample project, available at:

   bitbucket.org/gaudrito/alchemist-example

7. Import the downloaded folder as a Gradle project.

   - **Eclipse:** click on `File > Import`, then select the type `Gradle > Gradle Project`. Click on `Next` twice, then select the downloaded folder through `Browse` in the upper right corner. Complete the procedure by clicking on `Finish`. If errors appear, you may also need to right-click on the newly created project and then click on `Gradle > Refresh Gradle Project`.
   - **IntelliJ:** click on `File > Open` then select the `build.gradle` file within the downloaded folder. Click on `Open as Project` then compile the resulting form as in the following:


Import Project from Gradle

Gradle project: ~/repos/tech-alchemist-example

☐ Use auto-import
☐ Create directories for empty content roots automatically

Group modules: [✓ using explicit module groups] using qualified names

☑ Create separate module per source set

Use default Gradle wrapper (recommended)
[✓ Use Gradle 'wrapper' task configuration] Gradle wrapper customization in build script
Use local Gradle distribution

Gradle home:

Gradle JVM: Use Project JDK (java version "1.8.0_131", path: /Library/Java/JavaVirtual...k1.8.0_131.jdk/Contents/Home)

Project format: .idea (directory based)

Global Gradle settings

Cancel   OK

2 The sample project

2.1 Run the simulations

In order to execute the project, you need to create a new *run configuration*.

- **Eclipse:** click on `Run > Run Configurations`, select type *"Java Application"* then click on the *"New"* button (in the upper left corner). In the *"Main"* tab enter the following:

  Project: the sample project

  Main class: `it.unibo.alchemist.Alchemist`

  In the *"Arguments"* tab enter the following:

  Program arguments: `-g src/main/resources/example.aes`
  `-y src/main/resources/protelis.yml`

  You can then click on `Run` to launch the Alchemist graphical interface.

- **IntelliJ:** click on `Add Configuration > + > Application` and then set:

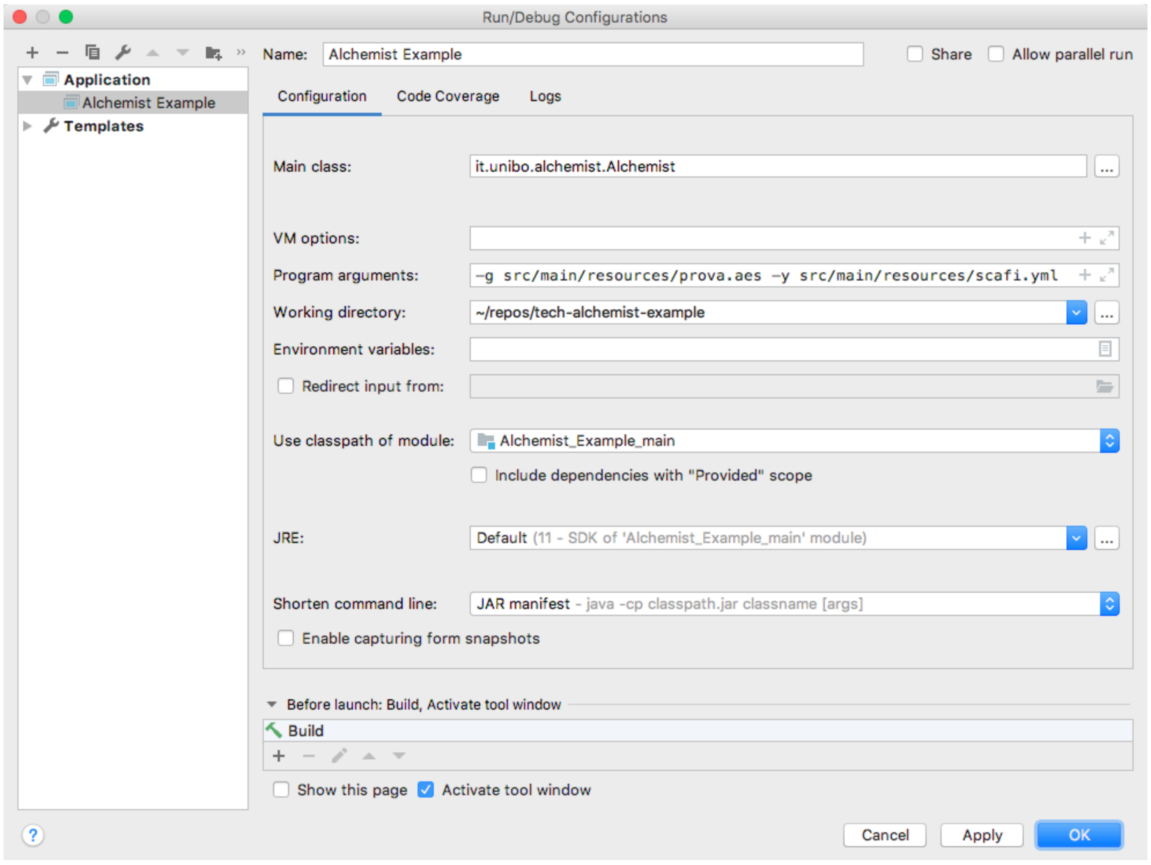
  Main class: `it.unibo.alchemist.Alchemist`

  Program arguments: `-g src/main/resources/example.aes`
  `-y src/main/resources/scafi.yml`

  Use classpath of module: `Alchemist_Example_main`

  Shorten command line: `JAR manifest`

  Press `OK` and then the green play button. The window should resemble the following:



Run/Debug Configurations

Name: Alchemist Example

Main class: it.unibo.alchemist.Alchemist

VM options:

Program arguments: -g src/main/resources/prova.aes -y src/main/resources/scafi.yml

Working directory: ~/repos/tech-alchemist-example

Environment variables:

Redirect input from:

Use classpath of module: Alchemist_Example_main

Include dependencies with "Provided" scope

JRE: Default (11 - SDK of 'Alchemist_Example_main' module)

Shorten command line: JAR manifest - java -cp classpath.jar classname [args]

Enable capturing form snapshots

Before launch: Build, Activate tool window

Build

Show this page ✓ Activate tool window

- **Command line:** it is also possible to launch the simulations without an editor, using the commands:

```
./gradlew runProtelis   gradlew.bat runProtelis
./gradlew runScafi      gradlew.bat runScafi
```

  depending on whether you want to run Protelis or ScaFi, and on whether your platform is Unix/Linux/MacOS or Windows.

Once the Alchemist graphical interface is ready, you can:

- press `P` to play/pause the simulation;
- press `L` to toggle display of links between devices;
- press `M` to toggle display of the marker;
- mouse scroll wheel to zoom in/out and drag to move the viewport;
- double click on a node to view additional details on its contents;
- change the graphical effects (upper left side of the window).

2.2 Inspect and modify the project

In the project explorer (left side) you can inspect the folder structure of the sample project. Expand the needed folders to find the main files of this project:

- `src/main/resources`
  - `protelis.yml`: the description of the Protelis simulation scenario.
  - `scafi.yml`: the description of the Scafi simulation scenario.
  - `example.aes`: the graphical effects displayed by the simulator interface.
- `src/main/protelis`
  - `example.pt`: the Protelis program to be run.
- `src/main/scala`
  - `example.scala`: the Scafi program to be run.

2.2.1 The program

The (commented) program contains few helper functions, including a function `elapsed` counting elapsed rounds, `constant` to obtain values that do not change across rounds, `ABF` computing the adaptive Bellman-Ford algorithm. The `main` function starts with two blocks setting the source and getting distance information respectively. The last blocks regulate the movement of devices: first by computing a `timeToGo` after which movement starts, and then a `target` towards which the device should move.

2.2.2 Simulation scenario

The simulation scenario contains few sections. First, some general information is set: the chosen language (Protelis or ScaFi), seeds for the random number generator, the network model (in this case, connection happens when the euclidean distance is smaller than 1).

In section `pools` are reported things to be linked later, describing the events to be executed (program execution, Brownian movement, moviment towards a target), together with a temporal distribution and other parameters.

Finally, in section `displacements` the devices are displaced according to geometric shapes (point, circle, rectangle), and are associated to events to be executed (and possibly custom data contents).

2.2.3 Graphical effects

The graphical effects file is not meant to be manually edited, but modified directly from the Alchemist graphical interface instead, clicking on the rectangular shape in the upper left side, close to the arrows, with a `DS` label on it. A dialog will appear, showing that the current effect is tuning node colors through the "molecule" `abf` (value exported into the simulator through `put`), together with parameters tuning the color palette.

3 Exercises

Edit the file `example.pt` or `example.scala`, incrementally, in order to compute the following in every device.

1. The number of neighbour devices.
2. The maximum number of neighbour devices ever witnessed by the current device.
3. The maximum number of neighbour devices ever witnessed by any device in the network.
4. Move towards the neighbour with the lowest number of neighbours.
5. Move away from the neighbour with the highest number of neighbours.
6. Move as if the device was attracted by the neighbour with the lowest number of neighbours, and repulsed by the neighbour with the highest number of neighbours.
7. Move as if the device was repulsed by every neighbour, and by the four walls of the rectangular box between points `[-4,-3]` and `[4,3]`.

3.1 Hints

- In the first few exercises, start by reasoning on when/where to use `nbr` ("collecting from neighbours") and `rep` ("collecting from the past").
- In order to move a device, you need to store a coordinate in the `target` variable of the simulator, with `env.put("target", ...)` in Protelis, or `node.put("target", ...)` in ScaFi. Part of `example.*` is already handling this, so you can just edit that part.
- You can use the defined function `getCoordinates()` to get the position of the current device. Coordinates can be composed as physical vectors in Protelis:

  ```
  [1,3] + [2,-1] == [3,2]
  [2,4] * 0.5    == [1,2]
  ```

  Similar operators are also defined in `example.scala`.
- In the last few exercises, you can model attraction/repulsion using the classical *inverse square law*. More precisely, if $\vec{v}$ is the vector between two objects, the resulting force is $\vec{v}/|\vec{v}|^3$ where $|\vec{v}| = \sqrt{v_x^2 + v_y^2}$.

4 Links

- The official Protelis webpage:

  `protelis.github.io`
- The official Scafi webpage:

  `scafi.github.io`
- The official Alchemist webpage:

  `alchemistsimulator.github.io`