LC2B_ User Manual

Maha BOUSSABBEH

14 octobre 2014

Table des matières

Table des figures

Chapitre

1 Software installation

LC2B requires Java 6 or greater to run properly. Please refer to appendix A for information about Java installation.

Once a recent Java Virtual Machine (JVM) is installed, ran LC2B.jar by double-clicking on it, or from the command line :

```
$ java -jar visidia.jar
```

Chapitre

2 Graphical User Interfaces

2.1 Global View

When *LC2B* starts, the main window appears(see Figure 2.1). The Graphical User Interface (GUI) consists in three parts :

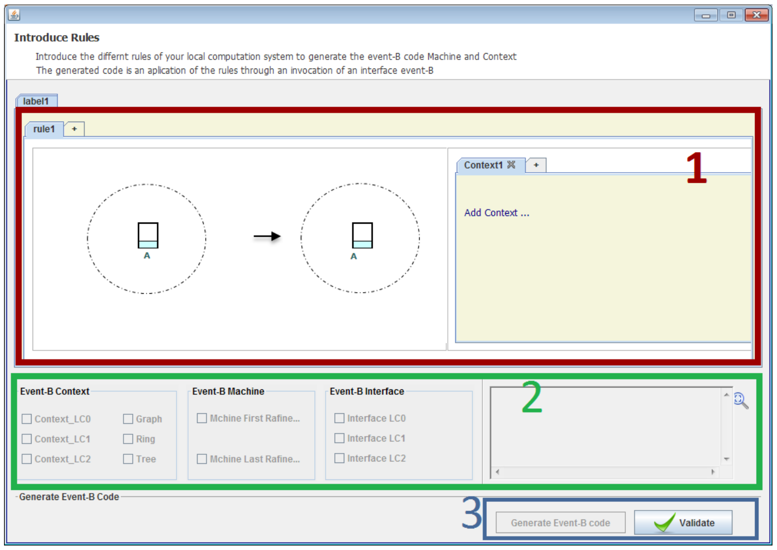
- At the top- center, a tabbed panel for introducing rewriting rules (a single tab for each rule)

- At the bottom center , a toolbar used for displaying the list of files to generate as well as the list of the introduced rewriting rules.

- At the bottom, a toolbar representing the most useful functionalities associated to the current view.

2.2 Set Properties

A network is represented by a graph whose nodes stands for processors and edges for (bidirectional) links between processors. The graph describes basic properties of the network on which the distributed algorithms are running. Formally, a network can be straightforwardly modeled as a connected, undirected and simple graph where nodes denote processors and edges denote point-to-point communication links. An undirected graph means that there is no distinction between the two nodes associated with each edge. A



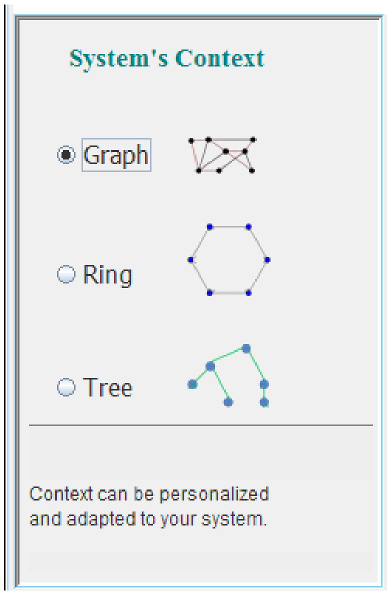
Introduce Rules

Introduce the differnt rules of your local computation system to generate the event-B code Machine and Context
The generated code is an aplication of the rules through an invocation of an interface event-B

Figure 2.1 – LC2B main Window

graph is simple, if it has at most one edge between any two nodes and no edge starts and ends at the same node ; it is obviously expressed by the choice of the relation representation by a set. A graph (directed or not) is connected, if for each pair of nodes, there exists a path joining these two nodes.

Basically, the network is represented ad a connected, simple and undirected graph. But



Context can be personalized and adapted to your system.

Figure 2.2 – Choose network properties

these properties can be extended to define a tree, ring, etc. Formal specifications of the network are generated automatically into an Event-B-Context as selected by the user (see Figure 2.2) :

- Choose the "graph" context to generate Event-B code of connected, simple and undirected graph. You can extend the generated context by adding other properties related to your network

- Choose the "tree" context to generate Event-B code of a network defined as a tree

- Choose the "ring" context to generate Event-B code of a network defined as a ring.

At every time, each node and each edge is in some particular state which is encoded by a node or an edge label. These labels allow nodes to perform an elementary step of computation according to some rewriting rules. As presented in Figure 2.3, you can introduce node labels as well as edge labels in a text-box in which you use a comma to separate each label.

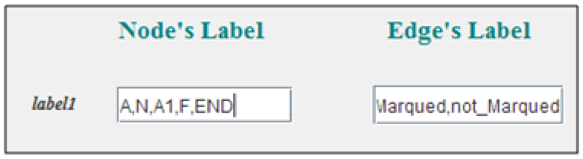


Figure 2.3 – Introduce Labels

2.3 Rewriting Rules

Rewriting rules can be defined thanks to the graphical interface, enabling the user to draw an algorithm only with the mouse.

When the main window appears you can create a new set of rules :

- Double click with left mouse button on dashed circle to add a node (and an edge linking this node to the central node

- Simple click with right button mouse on a node or an edge to edit its properties.
- Double click with left mouse button on a node to remove it
- Simple click with right button mouse on the close bottom of the tabbed panel to remove a rule.

One of three rules used to create a spanning tree with termination detection is presented in Figure 2.4. This rules means :
If u is a node of label "A" with a neighbour of label "A" with a marked edge between them, and u has no neighbour of label "N" with an unmarked edge ("Context 1"), then u can take the label "F". In fact, there is another forbidden context hidden under the "Context 2" tab.

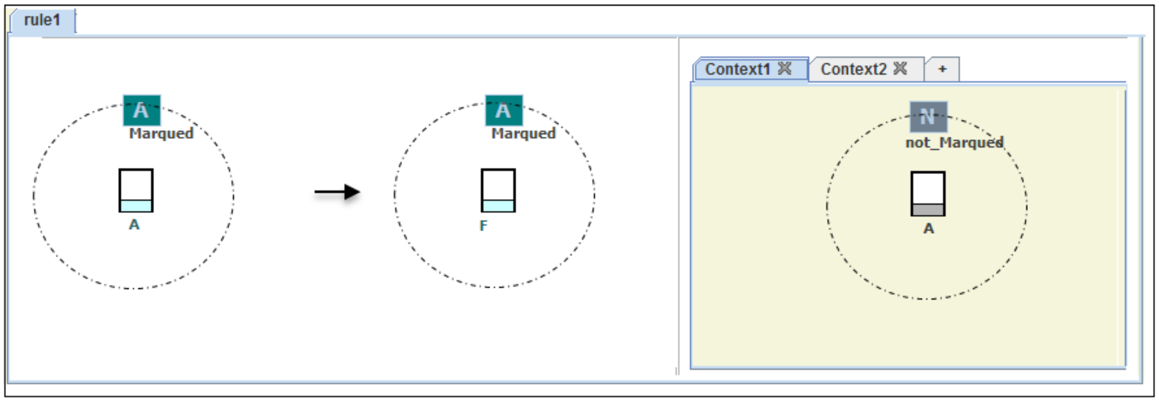


FIGURE 2.4 – Example of a rewriting rule Labels

2.4 Generating Event-B Files

Once you create all rewriting rules of your algorithm, you can validate and then generate the appropriate Event-B files. Before generation, you can see the different Event-B file names in a cheeked list-box. As presented in Figure 2.5 We distinguish three type of files :

- Event-B Context : we generate two Event-B contexts. A context containing generic properties of rewriting rules (LC0 properties or LC1 or LC2), and a context containing formal specifications of the network properties

- Event-B Machine : we generate Event-B machines specifying respectively the first and the last refinement level of the development. Computation steps of your algorithm are specified into the machine of the last level refinement.

- Event-B Interface : we generate the Event-B interface that we used while developing the computation steps of your algorithm. This interface contains generic properties of rewriting rule applications (LC0, LC1 or LC2).

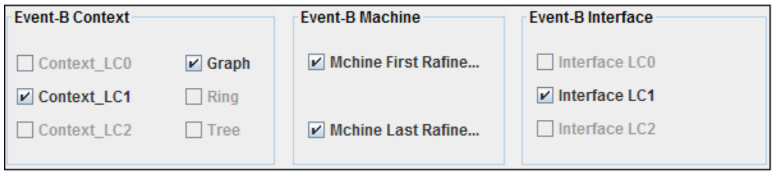


Figure 2.5 – Files to generate

Once you click on generate Event-B files, all files will be generated and added into your destination folder. You can then create a new Event-B project into the Rodin platform [1] and import the resulting Event-B files.

Important note You must have the modularization plug-in installed in your Rodin platform. You can obtain it through the Rodin menu *help/install New Software*. The plug-in instillation is detailed in the web site http ://wiki.event-b.org/index.php/Modularisation_Plug-in_Installation_Instructions

1. www.org.event-b

Annexe

A Install Java

You must have either a Java Runtime Environment (JRE) or a Java Development Kit (JDK) in version 6 installed on your computer. A JRE is enough to run LC2B.
If you do not have a JRE/JDK, of if its version is prior to version 6, please download and install JRE/JDK 6.
You can get the last official Sun JRE/JDK :

http ://java.sun.com/javase/downloads/index.jsp

or get an open source JDK, such as OpenJDK :

http ://openjdk.java.net/

Under linux Ubuntu, you can for example get openjdk-6 :
$ sudo apt-get install openjdk-6-jdk

Important notes : Depending on your operating system, you may encounter problems with some open source JRE/JDK. Please refer to section A.1.
You may need to adjust your JAVA_HOME environment variable, for example : $ export JAVA_HOME=/usr/lib/jvm/java-6-openjdk/
Under Windows, you may have to set the PATH environment variable to the JDK binaries.

A.1 Troubleshooting

A.1.1 GNU Java Compiler (GJC)

Please check that you DO NOT USE a JDK from the GJC. To check, run on command line :

```
$ java -version
$ javac -version.
```

If your system is configured to use GJC, please download and install another JDK, and use it as default.

A.1.2 OpenJDK

You may encounter problems using OpenJDK with Fedora distribution : Java does not execute.In this case, please install and use the official Sun JRE/JDK. Here is the recipe to install Sun JDK on Fedora.

* Get the last JDK version on http ://java.sun.com/javase/downloads/index.jsp Get the file with .bin extension, not the one with .rpm.bin extension.

* Intall the JDK (you need root privileges)

```
$ chmod a+x jdk-6u16-linux-i586.bin
$ mv jdk-6u16-linux-i586.bin /usr/lib/jvm/
$ cd /usr/lib/jvm
$ ./jdk-6u16-linux-i586.bin
$ alternatives –install /usr/bin/java java
    /usr/lib/jvm/jdk1.6.0_16/bin/java 3
  /usr/sbin/alternatives –config java
  (select 3 and enter)
```

* install the Java plugin for firefox (you need root privileges)

```
$ usr/sbin/alternatives –install /usr/lib/mozilla/plugins/libjavaplugin.so
    libjavaplugin.so
```

```
   /usr/lib/jvm/jdk1.6.0_16/jre/plugin/i386/ns7/libjavaplugin_oji.so 2
/usr/sbin/alternatives –config libjavaplugin.so
(select 2 and enter)
```