Choreographer Platfrom User Guide

N.V. Haenel, {valentin.haenel@gmx.de}

The Laboratory for Foundations of
Computer Science
School of Informatics
University of Edinburgh
2004 - 2005

latex

1 User Manual

This section describes the Choreographer Platform from the point of view of the user.

1.1 Introduction to the NetBeans Platform

This section gives an introduction to the features of the NetBeans Platform.

Diagram 1 shows a common screen layout for the Choreographer. Observe the Menus and Tool-bars at the top of the screen. The left hand side is occupied by the Explorer. The right hand side contains the Editor, and the bottom of the screen encompasses the output tab. One of the benefits of the NetBeans Platform is that all the visual components are float-able and can be repositioned within the main window, for example, diagram 2 shows an alternate layout.

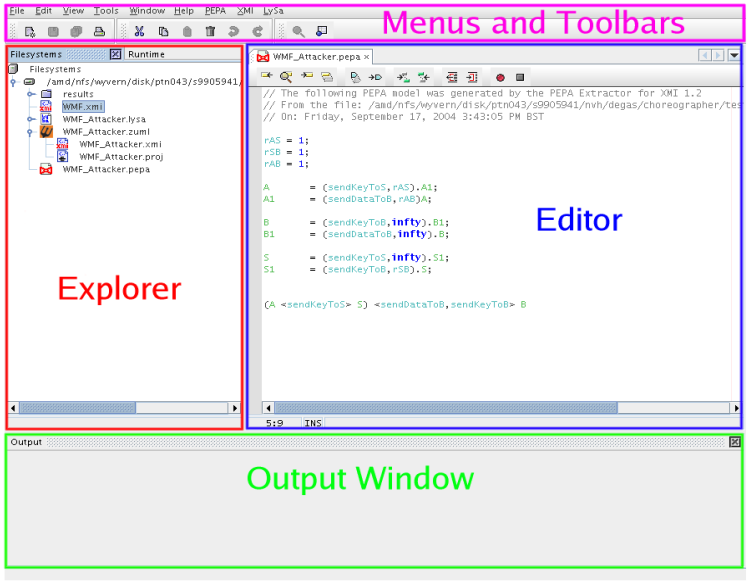

Figure 1: The Screen Layout of Choreographer

NetBeans 3.6 requires a directory to be mounted in order to view or use the files in it. To mount a local directory, right click the icon labelled `Filesystems` at the top of the explorer window. Select the option `mount`, and then `local directory`. Now select the desired directory using the file browser. Once

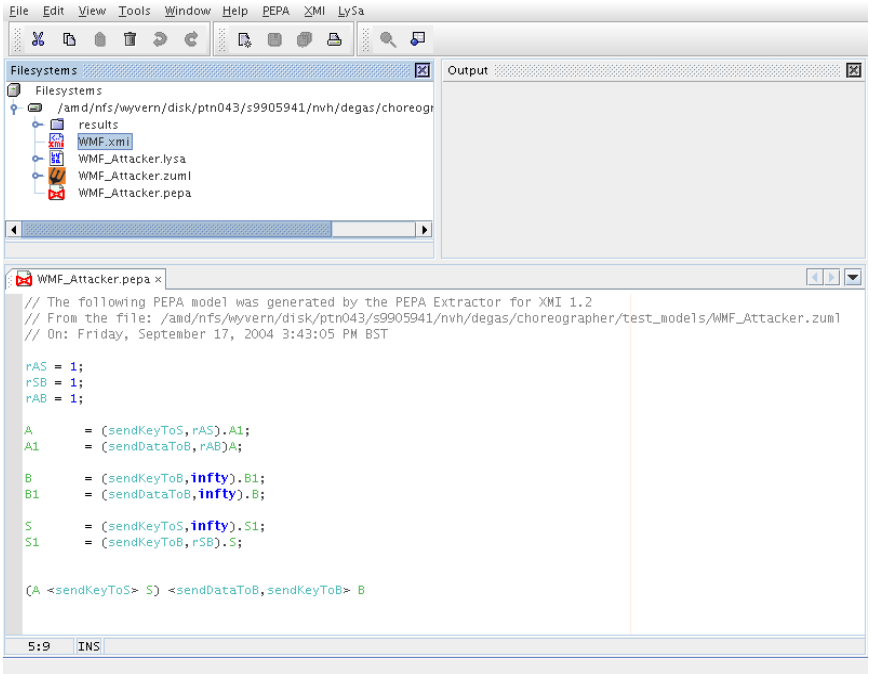

Figure 2: Alternative Screen Layout of Choreographer

mounted the directory is displayed as a child node of `Filesystems` and can be explored from there.

The Explorer is used to browse any mounted file systems, or local directories, and allows you to inspect their contents. Nodes in the explorer can be expanded and collapsed to vary the depth of the displayed file system. The Explorer recognises file types and will display an appropriate icon for each different file type. Furthermore all file types have an associated context menu that can be used to invoke actions specific to this file. Diagram 3 shows the features of the Explorer in detail.

The Menus and Tool-bars in Choreographer are used to invoke basic file and text operations such as `New File`, `Load File`, `Save File`, `Cut Copy`, `Paste`. `Undo Redo` and `Find`. Tool-bars can be enabled and disabled. Diagram 4 shows some of the more commonly used tool-bar items. All tool-bar items are equipped with tool-tips that appear when you hover the mopuse over the item for a couple of seconds.

The NetBeans Editor is a sophisticated text editor that can be used to view and modify the content of files. Diagram 5 shows the most important features of the Editor. Clearly visible, are one active and one inactive file in a tab, the editors own tool-bar, the main text window that is currently displaying a PEPA file, and lastly the field in the lower left hand corner that displays the line and column number. The tool-bar contains some useful text editing commands, such

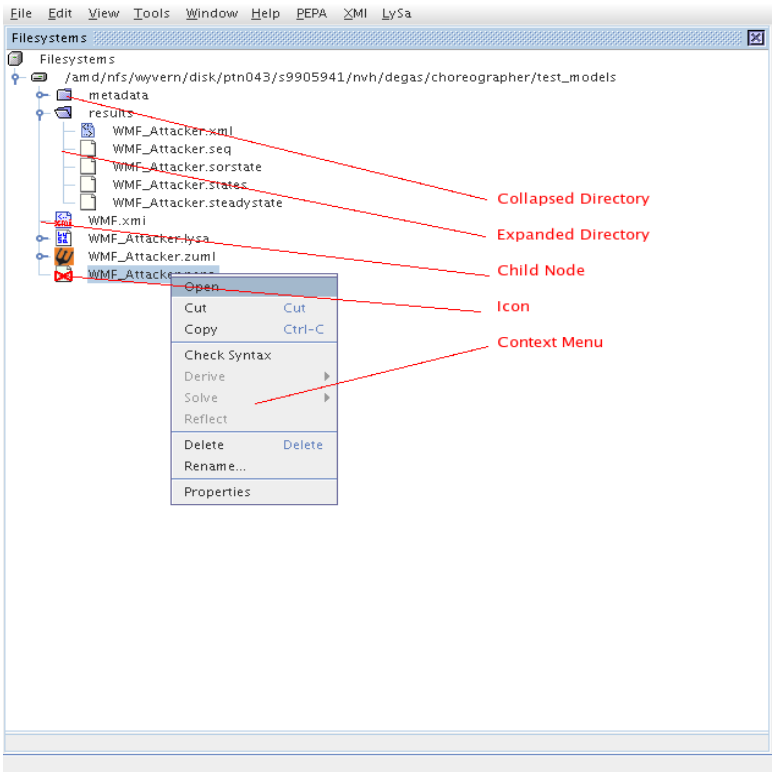

Figure 3: Details of the Explorer

as `Find Selection` and `Find Next Occurrence` that allow you to highlight a string within a document and then search the document for this string. All the tool-bar items have tool-tips that appear when you hover the mouse over the item for 2 seconds.

The editor itself can be customised in a variety of ways to make the user experience more pleasant and productive. Diagram 6 shows the multitude of configurable options and settings for the plain text editor. All the entries have associated tool-tips that are displayed at the bottom of the window. The properties window can be found under `Tools --> Options --> Editing --> Editor Options --> Plain editor`.

1.2 Installation Instructions

Choreographer itself has been developed in the Java programming language, the LySa tool is written in Standard ML(SML). Choreographer will happily work under both Windows and Linux. We used Red Hat Linux, kernel 2.4 and Windows XP(SP1) for development and testing. Choreographer requires:

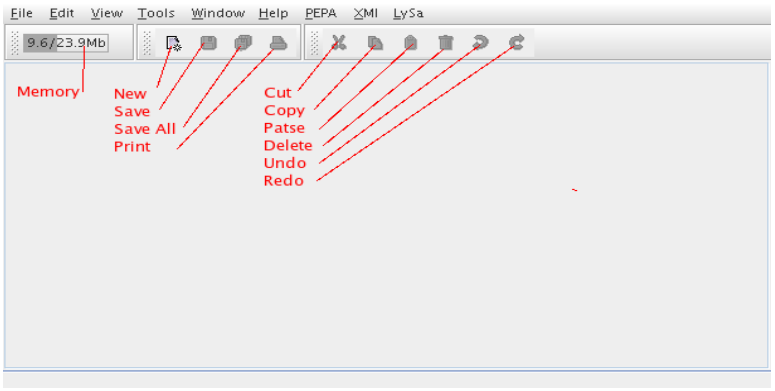

Figure 4: Menus and Toolbars

- An installation of Poseidon for UML, to draw the UML diagrams
- An installation of Java 1.4.2 Runtime Environment for the main application.
- An installation of Standard ML of New Jersey, for the LySa subcomponent.
- Download the latest .zip from the downloads section of the choreographer website at [1].
- Unzip the downloaded file.
- Launching:
  - For Windows, open the directory *choreographer* in an explorer and change to the `bin` directory. From there double-click *runidew*.
  - For Linux, open a terminal and cd to the *choreographer/bin* directory, and from there execute *./runide.sh*.
- Set the location of the SML runtime. Select`Tools --> Options --> LySa settings`, and in the properties window, select *SML Runtime*. To set the location click on the white box on the far right hand-side and select the appropriate file in the file browser window. (For Linux this is the file */bin/sml.bin* in the SML installation directory and for Windows it is the file *\bin\sml.bat* in your SML installation directory.)
- This completes the installation and you may now use the tool.

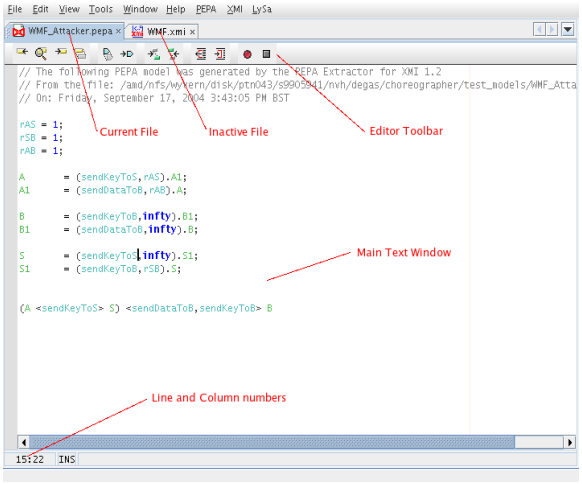

Figure 5: Editor Detail

1.3 Quick Start

- Download the WMF example from the choreographer website at [1]
- Select the *filesystems* icon in the explorer window on the left hand side of the screen. Click select `mount --> local directory`. Then select the directory that contains the example and expand it.
- Now select the file *WMF_Attacker.zuml* and right click on it. From the menu select either `Extract PEPA`, or `Extract LySa` to invoke the respective extractors. This produces the files *WMF_Attacker.pepa* and *WMF_Attacker.lysa* respectively, and you should see progress messages in the console at the bottom of the screen.
- To invoke the LySa tool, select the file *WMF_Attacker.lysa*, and right click. Then select `Invoke LySa` from the menu to run the tool. Any output appears in the console. For this example the tool will determine that the protocol has possible errors. This is the correct behaviour, as LySa models without errors provide no results that could be reflected.
- Reflection is now possible, right click on *WMF_Attacker.lysa* and select `Reflect`. You will now be presented with two file browsers, the first allows you to select the original file that contains the model, and the second allows you to enter a file name for the result of the reflection procedure. In the first file browser select the file *WMF_Attacker.zuml*, and in the second one leave the default as *WMF_Attacker.security.zuml*. This will produce the file *WMF_Attacker.security.zuml*, that contains the results from the analysis. The original file *WMF_Attacker remains intact.*

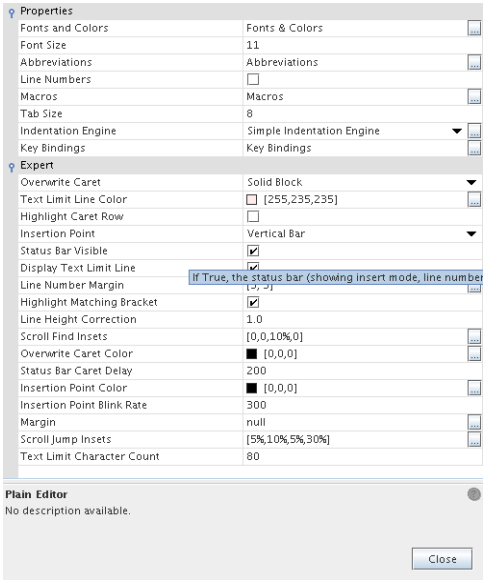

Figure 6: Editor Properties

- Now select the file *WMF_Attacker.pepa* and right click to show the menu. To solve the model:
  - `Select Check Syntax`
  - `Select Derive --> Silent`
  - `Select Solve --> SOR`

  Progress messages are displayed in the console as the tool advances.
- Reflection is now possible: right click on *WMF_Attacker.pepa*, and select `Reflect`. Again you will be presented with two file browsers, the first for selecting the original, and the second for selecting the destination file for the reflection. For the first, select *WMF_Attacker.security.zuml*, and then leave the default name as *WMF_Attacker.security.performance.zuml*. This produces the file *WMF_Attacker.security.performance.zuml* that contains the results from both the LySa reflection and the PEPA reflection.
- Lastly view the reflected model in Poseidon to convince yourself that it has worked.

1.4 Using the Choreographer

This section describes the three main modules offered by choreographer, namely PEPA, LySa and XMI, and what functionality is provided by each of these from

the users point of view, documenting what features are provided by each module and how to use them.

1.5 Support for the PEPA formalism

The PEPA module of choreographer is an interface layer that binds the PEPA Workbench, Java Edition [2] into the Choreographer framework. Support for the PEPA formalism is provided partly by the PEPA Workbench, and partly by the Choreographer platform.

The PEPA menu has the following layout:

- Check Syntax
- Derive
  - Silent
  - Compressed
  - Uncompressed
- Solve
  - LNBCG Solve
  - SOR Solve
- Reflect

These menu items are displayed in two distinct places, the PEPA menu in the menu-bar at the top of the application, and in the context menu for PEPA files in the explorer. PEPA files are displayed using a butterfly combinator icon in the explorer. Throughout the next section assume that `example.pepa` is the model being used.

The Choreographer provides some simple syntax checking that can be performed to assert the lexical and the syntactical correctness of the model. To invoke this from the menu select `Check Syntax`, the results of the check will be displayed in the output window.

The Choreographer provides State Space Derivation in three output formats, `Silent`, `Compressed` and `Uncompressed`. This will derive the state space of the model. `Silent` produces no output `Compressed` will produce a compressed version of the state space in the file *results/example.states*. This lists all the states but the sequential derivatives have been replaced with integer encoding. In addition a file *results/example.seq* is produced. This contains the integer encoding of the sequential derivatives and allows the compressed state space to be decoded. `Uncompressed` will produce the file *results/example.states*: this time the state space has not been encoded, and the file lists all possible states of the PEPA model in verbose detail. Progress messages will be displayed in the output window.

There are two available solvers for steady state solution, the linear biconjugate gradient method, and the successive over-relaxation method. The first is known to converge faster and use less memory and is therefore recommended. The steady state solution will be written to a file *results/example.steadystate* for the linear biconjugate gradient method, and *results/example.sorstate* for the successive over-relaxation. . To invoke the solvers, select either `Solve --> LNBCG Solve` or `Solve --> SOR Solve`. Progress messages will be displayed in the output window. The settings for the two solvers can be accessed through `Tools --> Options --> PEPA Module Settings --> (LNBCG/SOR) settings`.

`Reflect` is related to the Extractor/Reflector support offered by Choreographer and is described in 1.7

The Choreographer provides syntax highlighting for the PEPA formalism. This means that certain classes of identifiers, such as component and action identifiers, the PEPA keywords such as `infty`, operators such as `+`,all numerical values and comments are displayed in differing colours. This makes it easier and more comfortable to work with PEPA models.Syntax highlighting is enabled by default, and cannot be disabled. Diagram 7 shows the syntax highlighting. The syntax highlighting settings for the editor may be modified. Select `Tools --> Options --> Editing --> Editor Settings --> PEPA Editor.` This will provide you with a settings panel to tweak the settings for the PEPA editor. To change the default highlighting, select `Fonts and Colours`, this will open up a list of categories, such as `pepa-comment` or `pepa-keyword`. Each of these categories has a default font, size and colour that can be customised. Figure 8 shows the fonts and colours window for PEPA.

The Choreographer provides background parsing for PEPA files while they are being edited. This means that Choreographer will perform a syntax check at regular intervals while the file is being modified in the editor. To be precise: every time a single character modification to the file is made, a timer is restarted, when the timer expires,the automatic syntax check kicks in. If any syntax mistakes are found the line in question is underlined clearly with a red wave like graphic, and a red cross appears in the margin of the editor. If you now hover the mouse over the red cross, a short description of the syntax error is displayed. Diagram 7 shows a syntax mistake, the underline graphic and the short error message. The settings for the background parsing can be customised using `Tools --> Options --> PEPA Module Settings --> General`. This allows you to enable/disable background parsing and set the delay for the timer.

For more information about the State Space Derivation and the two solvers is included in the User Manual for the PEPA workbench that can be obtained from **??**.

.

1.6 Support for the LySa formalism

The LySa module offers support for the LySa formalism.

the LySa menu layout is as follows:

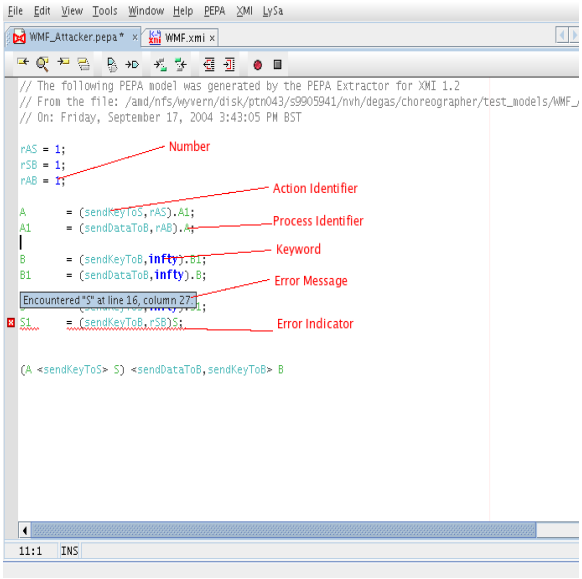

Figure 7: PEPA Editor Details

- Invoke LySa Tool
- Reflect

This menu is accessible from the LySa menu item in the menu-bar, and from the context menu for LySa files in the explorer.Lysa files are recognised by the explorer and displayed with the LySa icon.

The command `Invoke LySa` will run the LySa tool on the selected file. The Support for PEPA allows you to complete the model analysis step by step, i.e. check syntax, derive state space and solve model, and therefore provides a more elaborate menu.. The LySa support performs the equivalent steps for the LySa formalism with a single command. Thus invoking the LySa tool will perform a syntax check and also analyse the model.

The `Reflect` command is used in conjunction with the Reflector for LySa, and is described in section 1.7.

The support for the LySa formalism offers simple syntax highlighting, much th same way that it is provided for PEPA. The difference is that no background parsing is provided for LySa, and all syntax checking is be performed by the tool itself.

The editor and default syntax colours can be customised the same way they are customised for PEPA, however the settings are located in: `Tools --> Options --> Editing --> Editor Settings --> LySa Editor`.

There are a small number of settings to customise the LySa support offered by choreographer. `Tools --> Options --> LySa Settings`. This allows you

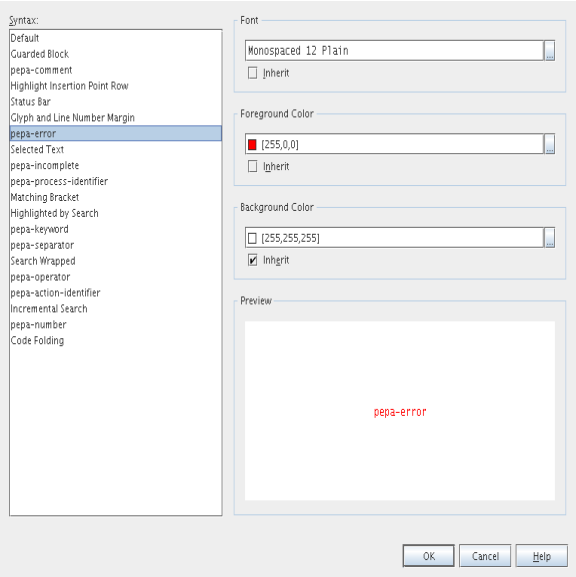

Figure 8: PEPA Editor Fonts and Colours Detail

to set the location of the Standard ML installation as described in the Quick-Start guide. There are two other options, `LySa Heap Image` and `number of instances, n`, however these are undocumented developer features used in debugging and development.

1.7 Support for the XMI format

Extractors Reflectors XMI etc. – explain what the extractors can do –extract reflect PEPA and LySa – explain how to invoke them – the imp details are discussed elsewhere

– the concept of Extractor/Reflector is discussed in background

References

[1] Degas choreographer website, university of edinburgh . `http://groups.inf.ed.ac.uk/choreographer/`.

[2] Pepa workbench, java edition, university of edinburgh . `http://homepages.inf.ed.ac.uk/s9905941/jPEPA/`.