Expert System Merl1n
version 2.1

Contents

About Merl1n

Expert system Merl1n is the application that defines knowledge based on rules and employs the inference engine with forward chaining. The main idea is that it is possible to define and check knowledge base (consisting of modules loaded based on actual demand) from Merl1n's expert system shell. Afterward the executable code is generated and distributed to the user in form of Java applet. The only requirement is to use Web browser with installed Java Plug-in version 1.4. In this way your company Guru can change knowledge and data on everyday basis and make news available to anybody connected to the Internet.

Installation

1. Install Java Runtime Environment (JRE) 1.4.0 and higher (http://java.sun.com).
2. Execute setup.exe. Select JRE's virtual machine from a list box as a default virtual machine.
3. In case of internationalization of the Merl1n copy font.properties.xx to the lib directory of JRE. Czech version of font.properties.cs is available in the Merl1n directory.

Installation consists of one stand-alone application that is Merl1n shell. There is also a .html document that demonstrates how to use Merl1n’s runtime module in form of Java applets.

Definition of Knowledge Base

Knowledge base consists of one or more modules and each module consists of the variable and rule definitions.

Variables

Variable types employed by Merl1n are following: *text*, *integer*, *real* and *boolean*. All of them except boolean can be declared as enumerative. It means that only specific values of the given type from a finite set can be assigned to them.

Example:
integer value
integer enumValue (1,3,7,9) /* declaration of the enumerative integer */

There is also a possibility to assign initial value to variable when it is declared. In case that the variable is enumerative then the initial value has to be one of declared values from enumerative set.

Example:
boolean tired := true
text disease (“flu”, “pneumonia”) := “flu”

A variable that has to be assigned by an user uses keyword *ask* followed by *associated text* in its definition. This identification moves the variable and its description into the dialog box that is displayed to the user before the rules of the module are evaluated. Only variables from the active module (being evaluated) are displayed. Variables from the other module are displayed when this module is loaded and it is about to evaluated. Based on that user is not pushed to assigned all variables at the beginning but only when appropriate module is required.

Example:
ask “Body Temperature” real temperature
ask “Are you tired?” boolean tiredness := false

Rules

Rules used by expert system Merl1n employ following pattern for their definition:

name*:*priority *if* condition *then* action *end*

Name identifies the rule, priority defines the order of the rule evaluation. It is an integer number. In case that priorities have the same values, rules are evaluated from up to down.

Condition part has to be evaluated to *true* or *false*. Expression in the condition part uses logical operators *and, or, not* and relational operators = (equal), != (not equal), < (less), <= (less or equal), > (more), >= (more or equal). Numerical expression uses standard operators +, -, *, / and %. Texts can be concatenated by +.

Action part of rule can do following:

- *variable* ***:=*** *value* assigns value to a variable
- ***print*** *“text”* prints text on the output window
- ***nl*** creates new line in an output text
- ***image*** *“description”* ***at*** *“url_ specification”* creates link to image in the output text
- ***document*** *“description”* ***at*** *“url_ specification”* creates link to html document in the output text
- ***stop*** stops the evaluation of rules
- ***exec*** loads and executes another module.

Action part can have one or more above action commands separated by comma (*,)*. Variables used by rules have to be defined before the rule definition. URL specification use standard pattern *protocol://host:port/filename* to locate images and .html documents. If there is no *protocol* and *host:port* specified then the file is searched in the directory relative to working directory of Merl1n's shell or runtime only in case that the *filename* does not start with slash (/). In opposite case the file is searched from the root directory.

Example:

```
healthy:10
if
        (temperature <= 37) and (tiredness = false)
then
        print “You are healthy!”,                                          /* Print output text */
        nl,                                                                /* Insert new line */
        image “Enjoy you life in Monte Carlo!” at “images/Monte_Carlo.jpg”, /* Insert link to image */
        stop                                                               /* No need to continue */
end

flu: 5
if
        (temperature > 38) and (tiredness = false)
then
        disease := “flu”,
        print “You have a “+disease, nl,                           /* Texts are concatenated */
        document “What to do?” at “http://www.doctor.com/flu.html”,/* Link to html document */
        exec “Treatment”                                  /* Next module named Treatment is loaded */
end
```

Inference Engine

Inference engine is based on forward chaining. It means that all input data are defined by user at the beginning and after that rules are evaluated based on their priorities. If the action command *stop* is reached during this evaluation or all of the rules were used the inference engine is stopped. Rule cannot be evaluated twice during one run of the engine but it is possible to restart the inference engine and the whole process can be repeated as many times as needed by user. The expert system remembers the input data from the last run so the user can change only what is needed to be changed and it is not necessary to define all input data again.

Inferencing starts with the first (top most) module. When the inference engine hits action command *exec* appropriate module is loaded, user is asked to define input data for this module and its rules are evaluated. When all rules are evaluated inference engine returns back to the original module and continues with evaluation of its rules. In case that stop command is reached, no matter in what module, the inference engine is stopped. Of course, the loaded module can load again some additional module. Appropriate modularization of knowledge base can radically lower the number of input data that have to be defined by the user at a given moment.

Expert System Architecture

The expert system Merl1n consists of expert system shell that is a stand-alone Java application and a runtime module that is a Java applet. The shell is used to define and test knowledge base. When the knowledge base is defined it is saved in a form of the persistent object and published through Java applets on the Internet. This runtime module requires installed Java Plug-in 1.3 (and higher) or Web browser that supports Java2 platform. Figure 1 describes the whole situation:

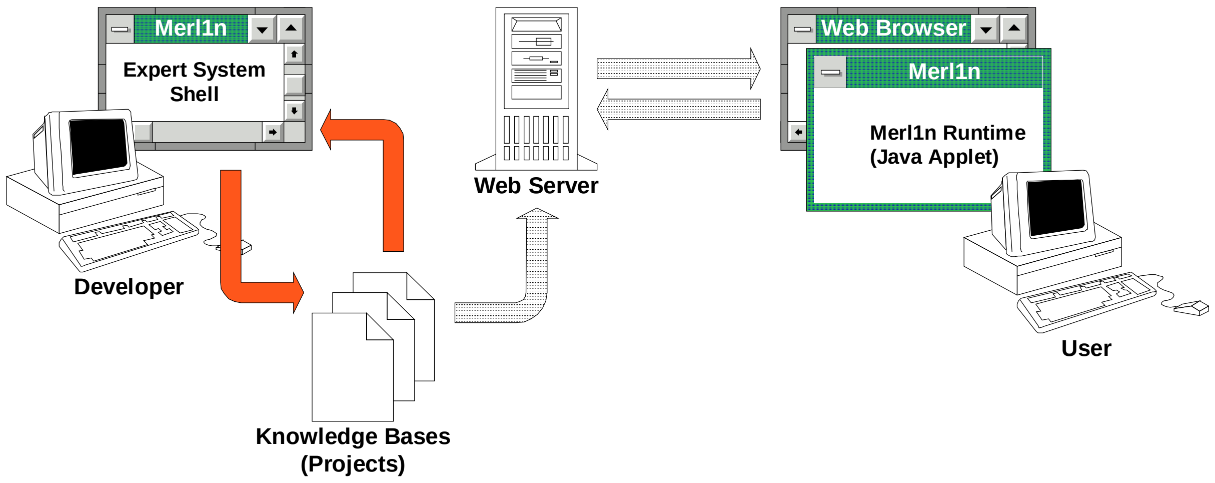
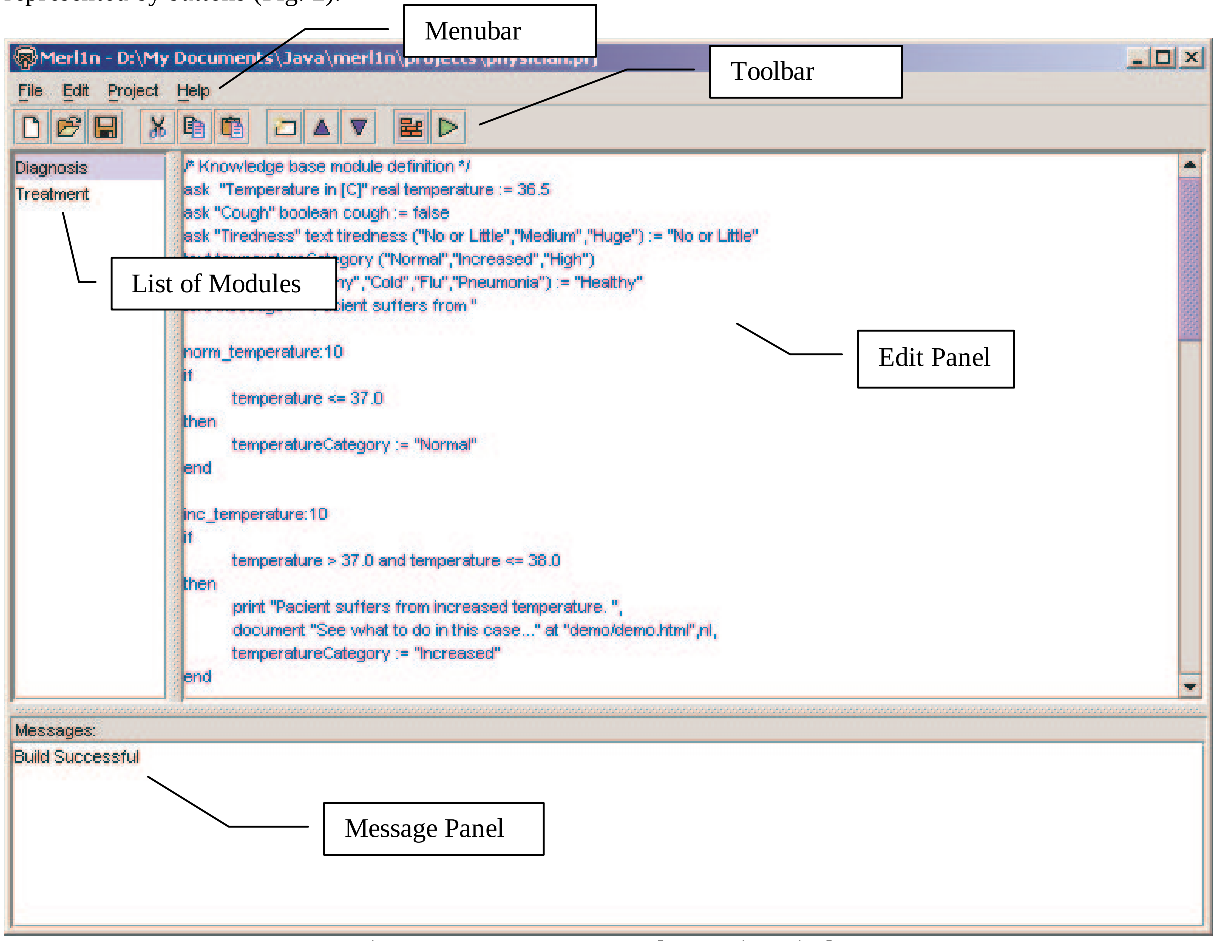


Fig. 1: Expert System Merl1n Architecture

ES Shell (Expert System Shell)

Expert system shell is a stand-alone application used for purposes of defining the knowledge base (project) and consulting with it. The life cycle of the project starts with the definition of modules, after that these modules are compiled what means that the project is built and ready for the consulting purposes. The order of module is important. If the variable is declared in one module and used in the second one then the first module has to be compiled first. Built project can be also published on the Internet and used by Merl1n clients implemented in form of Java applets.

Main window of ES Shell consists of three panels, a menubar and a toolbar with the most common commands represented by buttons (Fig. 2).



```
/* Knowledge base module definition */
ask  "Temperature in [C]" real temperature := 36.5
ask "Cough" boolean cough := false
ask "Tiredness" text tiredness ("No or Little","Medium","Huge") := "No or Little"
[illegible]gory ("Normal","Increased","High")
[illegible]ny","Cold","Flu","Pneumonia") := "Healthy"
[illegible]ient suffers from "

norm_temperature:10
if
        temperature <= 37.0
then
        temperatureCategory := "Normal"
end

inc_temperature:10
if
        temperature > 37.0 and temperature <= 38.0
then
        print "Pacient suffers from increased temperature. ",
        document "See what to do in this case..." at "demo/demo.html",nl,
        temperatureCategory := "Increased"
end
```

```
Build Successful
```

Fig. 2: Expert System Merl1n Main Window

List of Modules shows what modules are used and what is the order of how they are compiled into the final project. There is also a possibility to import such modules from the externally defined text files (menu command *File/External/Import Module...*). On the other hand it possible to export module into the text file (menu command *File/External/Export Module...*) and use it in another project. The order of module can be changed by appropriate commands (*Edit/Move Module Up* or *Edit/Move Module Down*).

Edit Panel enables to edit content of module selected in the list of modules. When the project is built and the content of the module is changed it is necessary to build the whole project again. Edit panel implements all standard rules for editing common for all standard text editors.

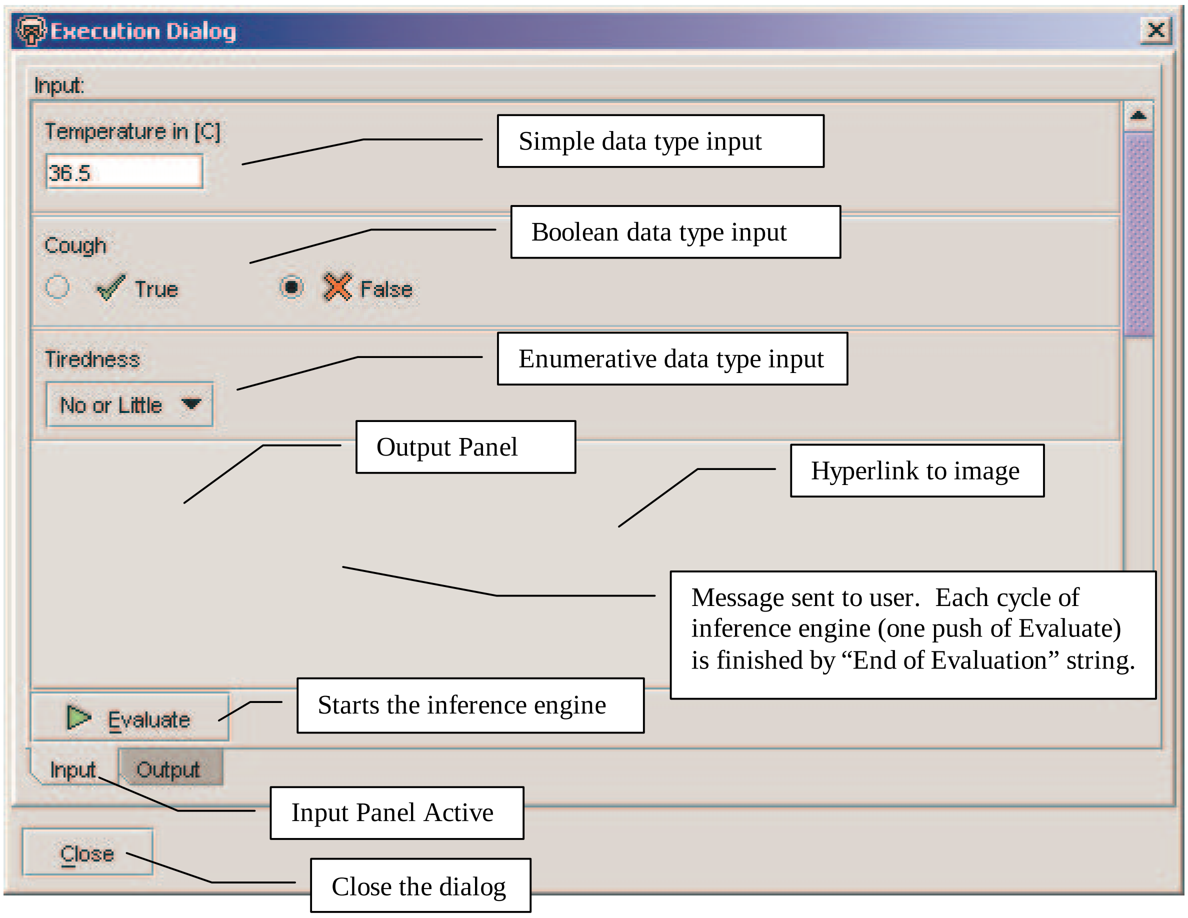
Message Panel serves as the output device where system messages are shown to user. If the knowledge base module is not correctly defined error message is sent to this panel during the process of compilation. At the same time the line that contains syntax error is highlighted in the edit panel.

Basic Commands

Command	Description
🗋	**New project** command clears the shell environment and initializes new project with the empty knowledge base module *Noname*.
📂	**Open project** command opens and loads already existing project from a file.
💾	**Save project** command saves the actual project on a disk. If the project is not built then it cannot be used by Merl1n Runtime. This is possible only with the project that was successfully built before it was saved.
✂	**Cut text** cuts highlighted text.
🗐	**Copy text** copies highlighted text..
📋	**Paste text** pastes text from a clipboard to the edit panel.
🗀	**Insert module** command creates new knowledge base module and inserts it right after the selected one.
▲	**Move module up** command moves selected module one step up.
▼	**Move module down** command moves selected module one step down.
🧱	**Build application** command compiles all modules and builds application that enables to consult with the knowledge base.
▷	**Execute application** command starts the consulting process with the knowledge base.

Consulting

Consulting with the expert system is enabled by inference engine that starts to evaluate rules of the first (in a list) module and asks user to enter all its input data. In case that additional module is loaded then a new dialog window is open and user is asked to enter new input data. The execution dialog window consists of two: input and output panels (Fig. 3,4).

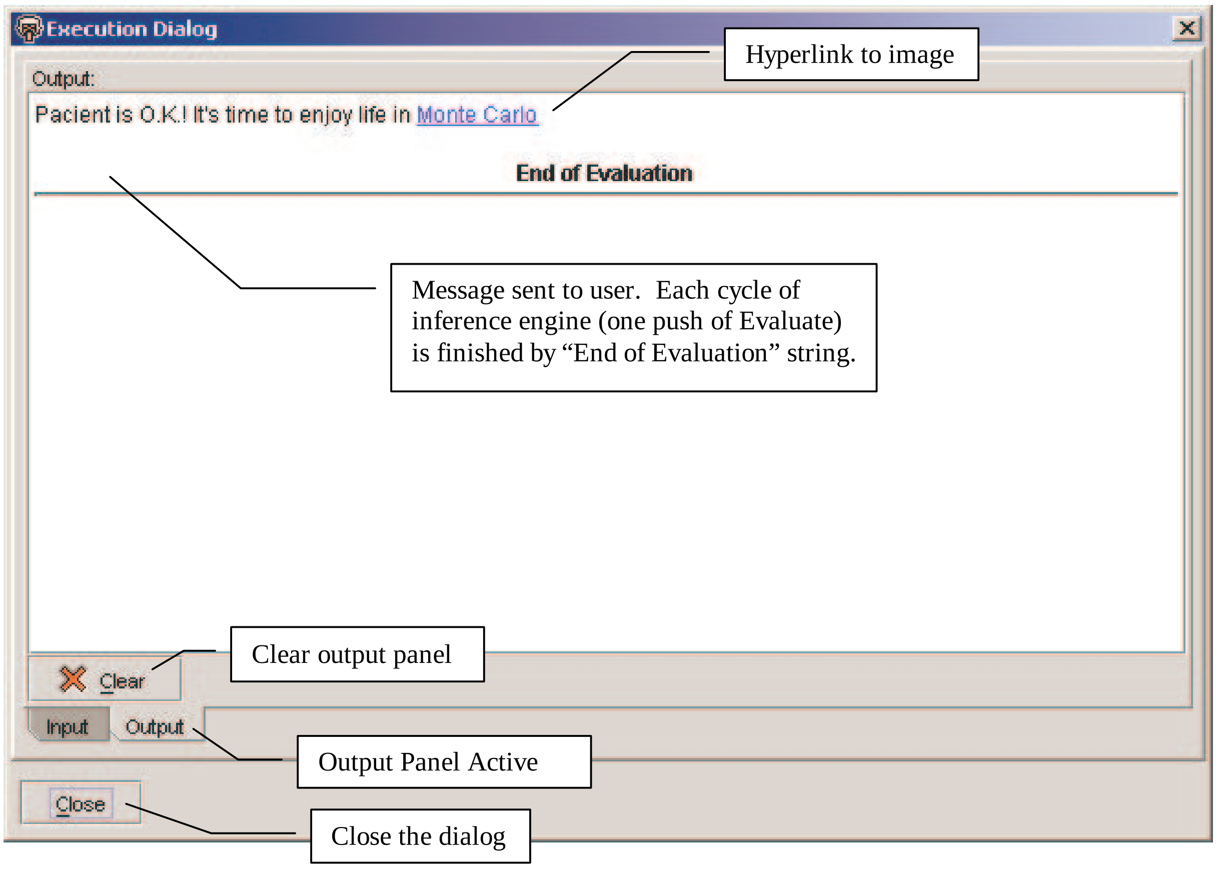

Execution Dialog
Input:
Simple data type input
Boolean data type input
Enumerative data type input
Output Panel
Hyperlink to image
Message sent to user. Each cycle of inference engine (one push of Evaluate) is finished by "End of Evaluation" string.
Evaluate
Starts the inference engine
Input
Output
Input Panel Active
Close
Close the dialog

Temperature in [C]: 36.5

Cough: True [✓ False]

Tiredness: No or Little

Fig. 3: Expert System Merl1n Main Window – Input Panel



Pacient is O.K.! It's time to enjoy life in Monte Carlo

Fig. 4: Expert System Merl1n Main Window – Output Panel

Each new cycle of inference engine uses the values of data defined in a previous run. In case that there is a need to reset data to the initial values defined in a knowledge base it is necessary to build the project again.

Merl1n Web Client

The purpose of Merl1n web client implemented in form of Java applets is to make all projects available to people who need an advice. The code for this runtime module is implemented in package m1web.jar. To execute m1web.jar applet the following must be specified within an .html file:

```
<OBJECT
  classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
  WIDTH = 350 HEIGHT = 30
  codebase="http://java.sun.com/products/plugin/autodl/jinstall-1_4-win.cab#Version=1,4,0,0">
  <PARAM NAME = CODE VALUE = merl1n.app.web.run >
  <PARAM NAME = ARCHIVE VALUE = m1web.jar >
  <PARAM NAME="type" VALUE="application/x-java-applet;version=1.4">
  <PARAM NAME="scriptable" VALUE="false">
  <PARAM NAME = "look&feel" VALUE   ="-x">
  <PARAM NAME = "label" VALUE   = "Physician Example">
  <PARAM NAME = "project" VALUE   ="physician.prj">

  <COMMENT>
     <EMBED
      type="application/x-java-applet;version=1.4"
      CODE = merl1n.app.web.run
      ARCHIVE = m1web.jar
      WIDTH = 350
      HEIGHT = 30
      look&feel = "-x"
      label =  "Physician Example"
      project = "physician.prj"
      scriptable=false
      pluginspage="http://java.sun.com/products/plugin/index.html#download">
         <NOEMBED>
         </NOEMBED>
     </EMBED>
  </COMMENT>
</OBJECT>

<!--
<APPLET CODE = merl1n.app.web.run ARCHIVE = m1web.jar WIDTH = 350 HEIGHT = 30>
<PARAM NAME = "look&feel" VALUE   ="-x">
<PARAM NAME = "label" VALUE   = "Physician Example">
<PARAM NAME = "project" VALUE   ="physician.prj">
</APPLET>
-->
```

A parameter "look&feel" specifies how the application would look: -x means cross-platform, -s means look default to the platform where the applet is executed. Parameter "label" defines the title of the applet window and finally parameter "project" defines the name of the file that contains built project saved from Merl1n ES Shell. This file is downloaded from the web server to the applet through the network.

Contacts

Ivo Vondrak
Cs. exilu 537
708 00 Ostrava – Poruba
Czech Republic

E-mail: ivo.vondrak@vsb.cz
Phone: +420 69 693 2160
Mobile: +420 602 718 027