Università degli Studi di Bari
Dipartimento di Informatica

LACAM
Machine Learning

Embedding CLIPS

Antonio Vergari

May 25, 2016

Embedding CLIPS in Java

CLIPSJNI

To host CLIPS from Java code one can use[1] the CLIPS Java Native Interface, whose latest version is beta 0.5.

In order to properly install it one has to make sure to have:

- the compiled library (which is a `.dll` file under Windows, `.so` for Linux, and `.jnilib` for OS X)
- the proper CLIPSJNI.jar file containing the lib headers, sources and a compiled version of the interpreter

To correctly write and run a Java program calling CLIPS routines, you have to:

- compile the code by putting in the classpath the CLIPSJNI.jar file
- run the compiled classes by putting again the CLIPSJNI.jar in the classpath and specifying the library path through the attribute `java.library.path`.

[1] Alternatives like Jess allow foreign languages to communicate both directions.

CLIPS JNI library

With the downloaded version come the pre-compiled binaries for OS X (libCLIPSJNI.jnilib) and for 32 and 64-bit version of Windows (CLIPSJNI32.dll CLIPSJNI64.dll). For linux users one has to compile it by itself (sigh). To do so: enter the library-src folder and execute[2]

```
1  make -f makefile.linux
```

Save the library under a meaningful path and remember it. Optionally one can specify it globally by putting it under Java lib dir.

[2] For different Linux distros one can experience different errors during compilation, please follow this link for common problem solving like setting the JAVA_HOME system var or the flags under 64 bits distributions

A simple sample project I

CLIPS framework can be accessed through one (or more) instance(s) of the object Environment. The callable methods of the object are the ways to access CLIPS interactive commands. Their names are self explanatory: clear, reset, assertString, eval,...
Now suppose our first Java program shall be a little REPL that takes a command from the console stdin, evaluates it by calling CLIPS eval functions and prints back to the console stdout the result.

```
import CLIPSJNI.*; // importing APIs

public class ClipsREPL {
  Environment clips = null; // Declaring an environment
  ClipsREPL() {
    clips = new Environment(); /* Instantiating a new environment */
    clips.clear(); /* clearing it */
  }
  ...
}
```

A simple sample project II

The main method implementing the REPL:

```
void repl() {
  boolean endInteraction = false;
  Scanner in = new Scanner(System.in);
  while(!endInteraction) {
    System.out.print("CLIPS>␣");
    String userInput = in.nextLine(); /* Read */
    try {
      String response = clips.eval(userInput).toString(); /* Eval */
      System.out.println(response); /* Print */
      if(response.equals("(exit)")){ /* Loop */
        endInteraction = true;}
  }catch(Exception e) {
    e.printStackTrace();}}}
```

Mind the use of the eval method of Environment, it throws a generic Exception (sigh). Potentially, you can call every CLIPS interactive command with that.

A simple sample project III

To compile our .java file in a .class, we have to specify where the CLIPSJNI.jar has been saved:

```
javac —classpath <path—to—CLIPSJNI.jar> ClipsREPL.java
```

To execute it, we need to specify a complete path[3]

```
java —cp .;<CLIPSJNI.jar—path> —Djava.library.path=<CLIPSJNI—lib—path> ClipsREPL
```

Have a look at the other included examples to see other uses.

[3] Remind to include all the needed path with **-cp**, even the current one if necessary. Path concatenation is done with ":" under UNIX and ";" on Windows.

A simple diagnostic classifier revised

Write a small Java program embedding a shorter version of the simple classifier to diagnose icterus diseases.

```
clips.load("icterus_simple.clp");
```

This time there are no rules prompting the user for observed symptoms, you have to gather them from Java (from the command line, for instance, or even from a GUI).

After you properly ask the user what the symptoms are, you have to assert in the WM the corresponding facts. **Hint:** use the assertString function of the Environment.

To apply inference you can simply call the Environment method **run**.

Instead, to retrieve a fact, for instance to check whether a diagnosis has been asserted, you can call the CLIPS defined function

```
PrimitiveValue fv =
    clips.eval("(get_all_facts_by_names diagnosis)").get(0);
String diagnosis = fv.getFactSlot("name").toString();
```