JPL Java API

Young W. Lim
3/14/14

based on the following document:
http://www.swi-prolog.org/packages/jpl/
http://professor-fish.blogspot.kr/2009/08/swi-prologs-java-interface-jpl.html

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice/OpenOffice.

JPL Java API (10A)

JPL (1)

```
Query q1 = new Query( "consult", new Term[] {new Atom("test.pl")} );

System.out.println( "consult " + (q1.query() ? "succeeded" : "failed"));

Query q2 = new Query( "child_of", new Term[] {new Atom("joe"),new Atom("ralf")} );

System.out.println( "child_of(joe,ralf) is " + ( q2.query() ? "provable" : "not provable" ) );

Query q3 = new Query( "descendent_of", new Term[] {new Atom("steve"),new Atom("ralf")} );

System.out.println( "descendent_of(joe,ralf) is " + ( q3.query() ? "provable" : "not provable" ) );

Variable X = new Variable();

Query q4 = new Query( "descendent_of", new Term[] {X,new Atom("ralf")} );

java.util.Hashtable solution;
solution = q4.oneSolution();
System.out.println( "first solution of descendent_of(X, ralf)");
System.out.println( "X = " + solution.get(X));

java.util.Hashtable[] solutions = q4.allSolutions();

for ( int i=0 ; i<solutions.length ; i++ ) {
    System.out.println( "X = " + solutions[i].get(X));
}
```

test.pl

```
child_of(joe, ralf).
child_of(mary, joe).
child_of(steve, joe).

descendent_of(X, Y) :-
    child_of(X, Y).
descendent_of(X, Y) :-
    child_of(Z, Y),
    descendent_of(X, Z).
```

JPL (2)

```
System.out.println( "each solution of descendent_of(X, ralf)");

while ( q4.hasMoreSolutions() ){
        solution = q4.nextSolution();
        System.out.println( "X = " + solution.get(X));
}

Variable Y = new Variable();

Query q5 = new Query( "descendent_of", new Term[] {X,Y} );

while ( q5.hasMoreSolutions() ){
        solution = q5.nextSolution();
        System.out.println( "X = " + solution.get(X) + ", Y = " + solution.get(Y));
}
```

test.pl

```
child_of(joe, ralf).
child_of(mary, joe).
child_of(steve, joe).

descendent_of(X, Y) :-
   child_of(X, Y).
descendent_of(X, Y) :-
   child_of(Z, Y),
   descendent_of(X, Z).
```

in Prolog

```
% This library is all we need to call Java from Prolog
:- ensure_loaded(library(jpl)).

main
:-
  % Create an object
  jpl_new(class([],['Test']), [], X),

  % Printing objects is like printing object ids
  write(X),nl,

  % Access a field of the object; happens to be static
  jpl_get(X, state, Y),

  % Prints whatever the state's value is
  write(Y),nl.
```

in Java

```
// This jar is all we need to call Prolog from Java
import jpl.*;

public class Test {

        public static int state = 0;

        public static void main(String[] args) {

                // Consult a Prolog file
                Query q1 = new Query("consult", new Term[] { new Atom("Test.pro") });

                // Prints boolean success/failure value
                System.out.println(q1.query());

                // Affect static field.
                // This way we can see whether Prolog sees the same JVM instance.
                Test.state = 42;

                // Invoke a predicate
                Query q2 = new Query("main");

                // Prints boolean success/failure value
                System.out.println(q2.query());
        }
}
```

Compile

javac -cp $CLASSPATH:/opt/local/lib/swipl-5.6.62/lib/jpl.jar:. Test.java
java -cp $CLASSPATH:/opt/local/lib/swipl-5.6.62/lib/jpl.jar:. -Djava.library.path="/opt/local/lib/swipl-5.6.62/lib/i386-darwin9.5.0" Test
true
@(J#0000000000016809476)
42
true

Server Applications

```
dispatch(AcceptFd) :-
        tcp_accept(AcceptFd, Socket, _Peer),
        thread_create(process_client(Socket, Peer), _,
                [ detached(true)
                ]),
        dispatch(AcceptFd).

process_client(Socket, Peer) :-
        setup_call_cleanup(tcp_open_socket(Socket, In, Out),
                           handle_service(In, Out),
                           close_connection(In, Out)).

close_connection(In, Out) :-
        close(In, [force(true)]),
        close(Out, [force(true)]).

handle_service(In, Out) :-
        ...
```

JPL Java API (10A)

References

[1] en.wikipedia.org
[2] en.wiktionary.org
[3] U. Endriss, “Lecture Notes : Introduction to Prolog Programming”
[4] http://www.learnprolognow.org/ Learn Prolog Now!
[5] http://www.csupomona.edu/~jrfisher/www/prolog_tutorial
[6] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
[7] www.cse.unsw.edu.au/~billw/dictionaries/prolog/negation.html
[8] http://professor-fish.blogspot.kr/2009/08/swi-prologs-java-interface-jpl.html
[9] http://www.swi-prolog.org/packages/jpl/