Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III
Prof. Dr. A. B. Cremers

Advanced Logic Programming
Summer Semester 2010
Dr. Günter Kniesel

Assignment 1

Due: Sunday, 25.04.2010, 23:59:59 via SVN

For help, contact alp-staff@lists.iai.uni-bonn.de (staff only) or alp-course@lists.iai.uni-bonn.de (staff and participants).

Please start working on the exercises early enough so that you can contact us in time in case of problems. Don't expect us to be available during weekend!

Program Transformation Background: *Normalization*

"Normalization" transforms a program into a simpler form that eases subsequent analyses or transformations. An often used normalization is elimination of chained and nested method invocations or field accesses. It introduces temporary variables to which intermediate results are assigned, so that the remaining expressions are free of nesting and chaining:

Chained or nested	Normalized
x = a.b.c; // chained field access	t1=a.b; x = t1.c;
x = f().b; // chained field access	t1=f(); t2 = t1.b;
f(g()); // nested invocation	t1=g(); f(t1);

A prerequisite for normalization is the identification of places in the program that need to be normalized. Expressed in terms of the Program Element Facts (PEFs) facts created by JTransformer, normalization must happen in the following cases:

I. *Chained field access:* a field access (`getFieldT`) has a "receiver" that is neither an identifier (`identT`) nor the atom `null`.
II. *Nested invocation*: An invocation (`callT`) has a "parent" that is neither an assignment (`assignT`) nor an expression statement (`execT`).

Task 1. *Understanding PEF documentation* (4 Points)

Read the documentation of the PEFs mentioned in the above background section. See http://sewiki.iai.uni-bonn.de/research/jtransformer/java_pef_overview. For an explanation of the notation see http://sewiki.iai.uni-bonn.de/research/jtransformer/notation.

Then answer the following questions:

a) Which is the argument position of the "receiver" argument in a `getFieldT`?
b) Which is the argument position of the "parent" argument in a `callT`?
c) Is there any common structure that all of the above-mentioned PEFs share?

Task 2. *Using JTransformer* (4 Points)

a) Load the JHotDraw project into your Eclipse workspace and assign it a factbase as described at http://sewiki.iai.uni-bonn.de/research/jtransformer/stepbystep
b) Run the query "?- classT(Id,Pkg,'DrawApplet',Members)." as described in the tutorial and write down the values that you get for Id, Pkg and Members.
c) If you are in exercise group *N* display the *N*-th element of the Members list in the editor (via the context menu item "Show in Editor"). Copy the highlighted source code of this element as the answer to this task.
d) Show the same element in the Factbase Inspector and write down how many subelements it has.

Task 3. *Normalization* (5 Points)

Create a project in your workspace containing a file "normalization.pl". In this file, write

a) a predicate "chained_access(Id)" that succeeds for all identities of fields for which the Normalization condition I is true.
b) a predicate "nested invocation(Id)" that succeeds for all identities of invocations for which the Normalization condition II is true.

Task 4. *Consulting files and running queries* (4 Points)

a) Consult your "normalization.pl" file into the factbase for the JHotDraw project.
b) Try out your "chained_access(Id)" predicate. Inspect the first two results in the editor and Factbase Inspector to verify that the predicate works as expected. Write down the two found source code snippets as the result of this task.
c) Do the same for the "nested invocation(Id)" predicate.

Tips

1. If your predicates do not work as expected, use the SWI-Prolog tracer to see what is going wrong. See http://sewiki.iai.uni-bonn.de/research/pdt/users/working_with_pdt for how to invoke and use the tracer.
2. Don't forget: All your source code and answers to questions must be submitted as files in the Subversion folder of your group.