Assignment 9

Due: Friday, 8.7.2016, 15:59 via Git

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

Start working on the exercises early enough so that you can contact your tutor in time if you have problems.

Submit your implemented predicates as a file named “assignment08/solutions.pl” in the Git repository of your group.

Add a file “assignment09/testRuns.txt” showing the console output of a session in which you test that **each** solution works for the provided input data and some queries that represent sensible test cases.

If no input data is provided in the text of the task, create some sensible input data.
If input data is represented as facts, include them into the “solutions.pl” file and add suitable comments.

Task 1. *Analyse terms* (7 Points)

Implement a predicate `term_type(+Term,-Type)` that unifies `Type` to the most specific type of `Term`. For instance, `?- term_type(1, T).` should unify T to ‘integer’, not to ‘number’ or ‘atomic’. The following test must succeed:

```
:-begin_tests(assignment_9_task_1).

test(term_type)  :-term_type(_, T0), assertion(T0=var),
                   term_type(a, T1), assertion(T1=atom),
                   term_type(1, T2), assertion(T2=integer),
                   term_type(1.0, T3),  assertion(T3=float) ,
                   term_type(f(1), T4), assertion(T4=compound),
                  X=f(X), term_type(X, T5),  assertion(T5=cyclic).
:-end_tests(assignment_9_task_1).
```

To run the test, add its code to your Prolog file, consult it and call?- run_tests.
The SWI-Prolog testing framework is documented in detail at http://www.swi-prolog.org/pldoc/package/plunit.html. See Section 2.2.5 One body with multiple tests using assertions for an explanation of assertion/1.

Submit your code (1 point per solved case) and the output of the successful test run (1 point).

Task 2. *Interactive IO predicates* (7 Points)

Implement a predicate `interactive_type_analysis/0` that performs the following steps:

1. Ask the user for a term to be analysed.
2. Read from the console (standard input) a term terminated by a period.
3. Analyse the term with the predicate from task 1.
4. Write the corresponding term type nicely as an answer to the console
5. (standard output). If you failed to solve task 1 just write 'This is the answer.'
6. Ask whether the user wants to continue or abort and tell him what to enter for which option.
7. Read the input and abort or start again from 1.

Submit your code (1 point per solved case) and the output of a successful test session in which you tested at least two terms before aborting (1 point).

Task 3. File IO predicates *(9 Points)*

Implement a predicate `general_type_analysis/0` that behaves like `interactive_type_analysis/0` but additionally

- asks the user at the start whether the session should also be logged in a file,
- asks for a file name (if the user answered 'yes') and writes all subsequent interaction (questions and answers) also to the file, up to and including the final 'abort' choice of the user,
- closes the file properly, no matter whether the interaction was terminated normally by entering 'abort' or abnormally by an exception or interrupt (like <Ctrl>-C).

Tip: Think first which type of IO (ISO, Edingburgh, SWI) is best suited for your task.

Submit your code (1 point per solved case).