1 Introduction to SWISH

Alternatively to the SWI-Prolog application, you can work with the local or online SWISH editor, which allows you to use the SWI-Prolog system with a different, more modern editor.

1.1 Starting local SWISH

In the PC rooms you can start SWISH by typing the command `swish` into a terminal. Next, minimize your terminal and open your browser on the local swish web page (`http://localhost:3050/`). The left side of this web page is the editor tab. You can create programs (or notebooks, which we will not use) here.

1.2 Programs

After creating a new program by clicking on the *Program* button, the editor opens the newly created program.

These programs are not saved to the hard disk permanently! If you want to save these programs, you will have to copy them to your own local files.

It invites you to type some prolog facts and rules into the editor. For your first program, enter the following facts.

```
father(anton,bart).
father(anton,daan).
father(anton,elisa).
father(fabian,anton).

mother(celine,bart).
mother(celine,daan).
mother(celine,gerda).
mother(gerda,hendrik).
```

Your program now contains `father/2` and `mother/2` predicates. The number after "/" indicates the number of arguments for that predicate, which is called the *arity*.

1.3 Queries

You can now ask queries about the prolog program. The SWI-Prolog system will then generate answers for your query, using the programs.

To query, use the bottom right panel in the swish web-page which invites you to type a query. It is indicated with a '?-' symbol.

Enter the following query (don't copy the '?-' symbol), and press enter to execute the query.

```
?- father(anton,X).
```

1.4 Results

The top right results panel will now show you Prolog's first answer to your query, namely `X = bart.`. You can either use one of the buttons to ask for the next answer(s), or press the ';' button on your keyboard to get the next answer. Continue doing so until Prolog has printed all answers.

1.5 Debugging

The debugger is your best friend when developing Prolog programs. Besides quickly tracing the mistakes in your program, it also allows you to step through your program like any other debugger. However, since Prologs execution scheme is a little more tricky, this also provides valuable insight into how your Prolog program is executed. **This is an essential development tool for Prolog - learn how to use it!**

By prepending `trace,` before your query, (e.g. `trace, father(anton,X).`) you can enable trace debugging for your query.

During each step you can perform a variety of actions, corresponding to the six buttons that show up in the results panel:

- Continue execution without tracing.
- Stepping to go to the next step in Prolog's execution.
- 'Stepping over' the current command (= execute the step without going deeper into the execution tree).
- Step out, to execute all subgoals of the current rule and continue execution afterwards.
- Retry to re-execute your last query. This is useful when you've performed a `skip` step and noticed your query failed and want to know why.
- Abort, to stop the query.

If you're not interested in debugging the entire execution of your program, but are interested in how some of your predicates or rules work, you can use breakpoints. To create a breakpoint, click on the gutter (where the line numbers are located) next to the line where you want to add breakpoint.

1.6 Final words

- During these exercise sessions we encourage asking questions when you're stuck somewhere. You can also always consult the SWI-Prolog manual at: `http://www.swi-prolog.org/pldoc/refman/`
- Active attendance is mandatory on all DT exercises.
- We very strongly recommend to first try to solve each exercise **individually**, and not to collaborate on a single computer. Designing solutions (instead of just programming them) is also part of this course, try to acquire this skill by thinking of your solutions yourself!
- The exam for this course has to be taken on the departmental computers, so use these exercise sessions to get some experience developing Prolog programs on these machines. There may not always be sufficient computers for everyone to use, in this case, try to use your own laptop.
- Structure your code! Use the layout shown below for your Prolog clauses to increase readability of your program. This layout is mandatory - if you write poorly formatted Prolog programs, points may be deducted from your grade.
  - For clauses with just a single body term:

    ```
    head_predicate :- body.
    ```

    or

    ```
    head_predicate :-
        body.
    ```

  - For clauses with multiple body terms:

    ```
    head_predicate :-
        body_1,
        body_2,
        ...,
        body_n.
    ```

Further style advice can be found in the article with guidelines on Toledo (under “`Documents`”).