Programming Paradigms
Unit 7 — Debugging and the Box Model

J. Gamper

Free University of Bozen-Bolzano
Faculty of Computer Science
IDSE
The box model of Prolog execution provides a simple way to show the control flow.

A box represents the invocation of a single predicate (procedure).

The box has four ports (with associated events):
- **CALL**: The first call of a predicate; control enters into the box.
- **EXIT**: The goal has been proven.
- **REDO**: The system comes back to a goal, trying to re-satisfy it, i.e., backtracking.
- **FAIL**: The goal/predicate fails.
Debugging

- The box model is used to **debug** the execution of Prolog programs.
- Predicate `trace/0` starts the **exhaustive tracing mode**
  - `notrace/0` stops the tracing mode.
- The debugger then displays a line for every port and waits for a command.
  - With `Return` or `c ("creep")` one steps to the next port.
  - The command `a` (abort) stops the execution of the query.
- Other debugger commands are available.
  - Usually displayed when entering `?` or `h`.
Consider the following facts (e.g., emil is the father of jan)

father(jan,emil).
father(julia,emil).
father(emil,arno).

The goal trace/0 activates "tracing"

?- trace, father(X,emil).
Call: father(_16,emil) ?
Exit: father(jan,emil) ?
X = jan ? ;
Redo: father(jan,emil) ?
Exit: father(julia,emil) ?
X = julia.
Now add the following rule

\[
siblings(X,Y) :- \text{father}(X, Z), \\
\quad \text{father}(Y, Z), \\
\quad X \neq Y.
\]

The box model for the goal \(\text{siblings}(X,Y)\) is nested

- The CALL of \(\text{siblings}\) enters the CALL of \(\text{father}\)
- The EXIT of \(\text{father}\) enters the CALL of another box for \(\text{father}\), etc.
The debugger output for the query trace, siblings(jan,Y)

(1) 0 CALL siblings(jan, Y) ?
(2) 1 CALL father(jan, Z) ?
(2) 1 EXIT father(jan, emil) ?
(3) 1 CALL father(Y, emil) ?
(3) 1 *EXIT father(jan, emil) ?
(4) 1 CALL jan \= jan ?
(4) 1 FAIL jan \= jan?
(3) 1 REDO father(Y, emil) ?
(3) 1 EXIT father(julia, emil) ?
(5) 1 CALL julia \= jan ?
(5) 1 EXIT julia \= jan ?
(1) 0 EXIT siblings(jan, julia) ?
X = julia ? ;
(1) 1 REDO: siblings(jan,julia) ?
(3) 2 REDO: father(julia,emil) ?
(3) 2 FAIL: father(Y,emil) ? c
(1) 1 FAIL: siblings(jan,Y) ? c
no
Remarks about the Box Model

- The exact form of the output depends on the Prolog system.
- The above output contains a **box number** in the first column and a **nesting depth** (call stack depth) in the second column.
- The asterisc "*" before EXIT marks that there are possibly further solutions (**nondeterministic exit**).
  - Otherwise, the box is already removed, and not visited during backtracking (i.e., no REDO-FAIL will be shown).
  - Because of such optimizations, the debugger output might violate the pure four-port model.
- Tracing is switched on by the predicate **trace/0** and switched off by the predicate **notrace/0**.
- Another useful debugging predicate is **spy/1**, which allows to specify specific subgoals, for which the user wants to obtain detailed information of the box model.
Spy Points/1

- Tracing is doing exhaustive debugging of all subgoals
- Another useful debugging predicate is `spy/1`
  - Allows to inspect the execution for selected subgoals only
- The following predicate sets a "spy point" on the predicate `father/2`
  
  ```prolog
  ?- spy(father/2).
  ```
- If the `debug` predicate is now used, Prolog executes the program without interruption until the first spypoint is reached
- Then one can continue debugging as with `trace` or "leap" to the next spy point (usually with the command `l`)
- `nodebug` stops the debugger, `nospy` removes the spy points
The debugger output for the query `debug, siblings(jan,Y)` after setting a spy point for `father/2`

```prolog
?- debug, siblings(jan,Y).
* Call: (8) father(jan, _G3270) ? creep
* Exit: (8) father(jan, emil) ? creep
* Call: (8) father(_G3159, emil) ? creep
* Exit: (8) father(jan, emil) ? creep
Call: (8) jan\=jan ? creep
Fail: (8) jan\=jan ? creep
* Redo: (8) father(_G3159, emil) ? creep
* Exit: (8) father(julia, emil) ? creep
Call: (8) jan\=julia ? creep
Exit: (8) jan\=julia ? creep
Exit: (7) siblings(jan, julia) ? creep
Y = julia.
```

Notice that the CALL port of `sibling` is not shown!
The comand leap jumps to the next spy point

?- debug, siblings(jan,Y).
* Call: (8) father(jan, _G1003) ? leap
* Exit: (8) father(jan, emil) ? leap
* Call: (8) father(_G889, emil) ? leap
* Exit: (8) father(jan, emil) ? leap
* Exit: (8) father(julia, emil) ? leap
Y = julia.