In the Lecture Series Logic Programming and Constraints

Software Development

Presented by Stéphane Bressan

Mode Declarations

- predicate1(+Arg1, ++Arg2, -Arg3, ?Arg4)
  - Arg1 is an input argument. It must be instantiated
  - Arg2 is an input argument and must be ground.
  - Arg3 is an output argument. It must be a variable
  - Arg4 is an input/output argument

:- mode predicate(+, ++, -, ?)

- Find examples in the documentation

Documentation

atom_length(+Atom, ?Length)

Succeeds if Length is the length of Atom.

+Atom Atom.
?Length Integer or variable.

Description
The length of an atom Atom is unified with Length. The length of an atom is the number of characters in the atom's name.

Fail Conditions
Fails if Length does not unify with the length of the atom Atom.

Resatisfiable
No.

Exceptions
(4) instantiation fault
Atom is not instantiated (non-coroutine mode only).
(5) type error
Atom is instantiated, but not to an atom.
(5) type error
Length is neither an integer nor a variable.

Examples
Success: atom_length(test, 4).
atom_length(test,L). (gives L = 4).
atom_length(as, X). (gives X = 2).
atom_length('4', 1).
Fail: atom_length(test, 5).
Error: atom_length(Atom, 2). (Error 4).
atom_length(Atom, 2.0). (Error 5).
atom_length(4, 1). (Error 5).
atom_length(as, 2.0). (Error 5).

Modules

```
:- module(example)
/* the module interface follows */
:- export p/5.
:- export q/2, op(q, xfx, 300).
/* the code of the module follows */
```


Export

- **Name/Arity**
  - procedure specification
- **domain(Spec)**
  - domain declaration
- **struct(Prototype)**
  - structure declaration
- **op(Prec,Assoc,Name)**
  - operator declaration
- **chtab(Char,Class)**
  - character class declaration
- **syntax_option(Option)**
  - syntax option setting
- **macro(Functor,Transformation,Options)**
  - macro (input transformation) declaration
- **portray(Functor,Transformation,Options)**
  - portray (output transformation) declaration
- **initialization(Goal)**
  - initialization goal specification

Modules

```
:- import(example).
test :- p(X, Y, Z, T, U).
```

Or

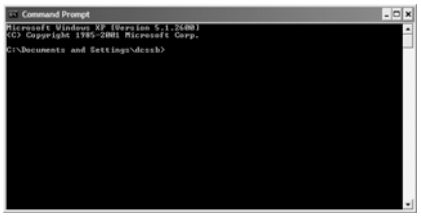
```
test :- example:p(X, Y, Z, T, U),
   example:(X q Y)
```

Modules

If the module is in a file and has to be compiled first, then use_module/1

:- use_module("/home/example").

DOS AND ECLiPSe



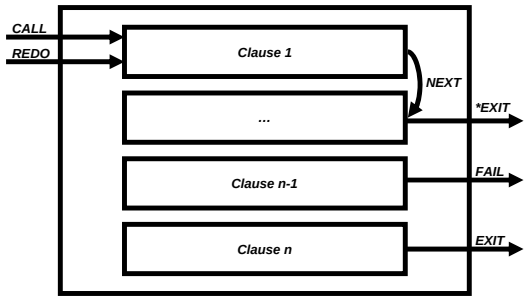
Compiling a File

- **compile(++File)**
  - Compile specified file or list of files File.
- Also [++File1, …, ++Filen]
- See also how to compile modules, import modules and use libraries

ECLiPSe Command Line Options

- −b bootfile
  - Compile the file bootfile before starting the session. Multiple -b options are allowed. The file name is expected to be in the operating system's syntax. The file is processed by ensure_loaded/1, i.e. it can be a precompiled file or a source file, and file extensions are added as specified there.
- −e goal
  - Instead of starting an interactive toplevel, the system will execute the goal goal. goal is given in normal Prolog syntax, and has to be quoted if it contains any characters that would normally be interpreted by the shell. The -e option can be used together with the -b option and is executed afterwards. Only one -e option is allowed.

Debugging: CALL, EXIT, REDO, FAIL and NEXT



CALL

- When a procedure is invoked, the flow of the execution enters the procedure box by its CALL port and enters the first clause box which could (since not all clauses are tried, some of them being sure to fail, i.e. indexing is shown) unify with the goal. It may happen that a procedure is called with arguments that make it sure to fail (because of indexing). In such cases, the flow does not enter any clause box. For each CALL a new procedure box is created and is given:
  - an invocation number that is one higher than that given for the most recent CALL port. This allows to uniquely identify a procedure invocation and all its corresponding ports.
  - a level that is one higher than that of its parent goal.
  - The displayed variable instantiations are the ones at call time, i.e. before the head unification of any clause.

EXIT

- When a clause of a predicate succeeds, the flow gets out of the box by the EXIT port. When a procedure exits non-deterministically (and there are still other clauses to try on that procedure or one of its children goals has alternatives which could be resatisfied), the EXIT port is traced with an asterisk (*EXIT). When the last possibly matching clause of a procedure is exited, the exit is traced without asterisk. This means that this procedure box will never be retried as there is no other untried alternative.
  - The instantiations shown in the EXIT port are the ones at exit time, they result from the (successful) execution of the procedure.

REDO

- When a procedure box is exited trough an *EXIT port, the box can be retried later to get a new solution. This will happen when a later goal fails. The backtracking will cause failing of all procedures that do not have any alternative, then the execution flow will enter a procedure box that an contains alternative through a REDO port. Two situations may occur:
  - either the last tried clause has called a procedure that has left a choice point (it has exited through an *EXIT port). In that case the nested procedure box is re-entered though another REDO-port.
  - Otherwise, if the last clause tried does not contain any non deterministically exited boxes, but there are other untried clauses in the procedure box, the next possibly matching clause will be tried.
- The last REDO port in such a sequence is the one which contains the actual alternative that is tried. The variable instantiations for all REDO ports in such a sequence are the ones corresponding to the call time of the last one.

FAIL, NEXT

- When a clause of a procedure fails the flow of the execution exits the clause box and leaves the procedure box via the FAIL port.
  - Note that the debugger cannot display any argument information at FAIL ports.
- If a clause fails and there is another possibly matching clause to try, then that one is tried for unification. The flow of the execution from the failure of one clause to the head unification of a following clause is traced as a NEXT port.
  - The displayed variable instantiations are the same as those of the corresponding CALL or REDO port.

Debugging with Tracer

Credits

Clipart and media are licensed from Microsoft Office Online Clipart and Media