

Loading the database (1)

Lecture notes 3.0

Running SWI-Prolog Creating truth tables

COMP 2411, session 1, 2004

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 1

Loading the database (2)

Suppose the previous three clauses are stored in the file
~/Programs/Prolog/plato.pl.

If you are in ~/Programs/Prolog/, here are two ways to
load plato.pl into the database:

```
?- consult('plato.pl').  
?- ['plato.pl'].
```

If you are in your home directory, here are two ways to load
~/Programs/Prolog/plato.pl into the database:

```
?- consult('Programs/Prolog/plato.pl').  
?- ['Programs/Prolog/plato.pl'].
```

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 3

The "swipl" command start SWI-Prolog.

A Prolog program consists of **clauses**, which can be either
facts or **rules**.

"person(socrates)." and "person(shane, warne)." are facts, while "mortal(X) :- person(X)." is a rule.

Clauses can be entered into the database directly from the
keyboard:

```
?- consult(user).          or      ?- [user].  
|: person(socrates).  
|: person(shane, warne).  
|: mortal(X) :- person(X).
```

followed by "Cntrl C" and "a" (for *abort*).

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 2

Loading the database (3)

You can omit the .pl extension.

Quotes are useless in case the argument contains no
special character (dots, slashes, etc.) and directory names
do not start with an uppercase letter.

For instance, if you are in ~/Programs/Prolog/, here are
two other ways to load plato.pl into the database:

```
?- consult(plato).  
?- [plato].
```

And of course, you can use absolute path names instead of
relative path names.

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 4

Listing the database (1)

You can type "listing." to list the content of the database, but the output then includes the **built-in predicates** that have been automatically loaded.

It is often more useful to list only specific **user-defined predicates**, of **any arity**:

```
?- listing(person).  
person(socrates).  
person(shane, warne).  
or of a specific arity:  
?- listing(person/2).  
person(shane, warne).
```

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 5

Queries (1)

After the program has been loaded, you usually want to ask **queries**:

```
?- mortal(socrates).  
Yes  
?- mortal(warne).  
No
```

If you hit carriage return before the query is complete (for instance, you might forget to type the final dot), continue typing on the next line:

```
?- mortal(warne)  
| .  
No
```

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 7

Listing the database (2)

You can also list rules:

```
?- listing(mortal).  
mortal(A) :-  
    person(A).
```

The fact "person(socrates)." is built from the **unary predicate** person.

The fact "person(shane, warne)." is built from the **binary predicate** person.

The rule "mortal(X) :- person(X)." is built from the **unary predicate** mortal.

Facts *and* rules can be built from the same predicate.

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 6

Queries (2)

A query can contains **variables**:

```
?- mortal(X).  
X = socrates ■
```

If you are happy with the **solution**, hit carriage return:

```
X = socrates  
Yes
```

If you want *alternative* solutions, type a semi-colon:

```
X = socrates ;  
No
```

No should be interpreted as *no more* solutions.

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 8

Modifying the database (1)

We can modify the database from the keyboard:

```
?- [user].
|: person(warne).
Warning: (user://1:14):
        Redefined static procedure person/1
```

Type "Cntrl C" and "a". Observe the changes:

```
?- listing(person).
person(warne).
person(shane, warne).
Yes
?- mortal(X).
X = warne ;
No
```

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 9

Modifying the database (3)

More generally, you type "make ." to update the database with the latest version of the files that have been loaded into the database.

Every time you update the database, either from the keyboard, or by typing "make .", or "consult(...).", or "[...].", you *redefine* all predicates that were already defined in the database: the new clauses that are built from a given predicate *replace* the clauses in the database that are built from the same predicate.

consult is the equivalent of reconsult in most other implementations of Prolog. (reconsult is not an SWI-Prolog built-in.)

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 11

Modifying the database (2)

If plato.pl was loaded into the database, modifying the database from the keyboard does not modify plato.pl.

Assuming that the database was loaded with plato.pl, if you edit plato.pl and add "person(warne)", typing "make." will update the database. Observe the changes:

```
?- listing(person).
person(socrates).
person(warne).
person(shane, warne).
Yes
?- mortal(X).
X = socrates ;
X = warne ;
No
```

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 10

Built-in editor

You can use the built-in editor by typing "emacs.". Navigate in the editor in order to find and load your file.

You can also edit plato.pl by typing "edit(plato).". (change "plato" by the full path name if you are not in the right directory), provided that you create a .plrc file in your home directory, edit it and write:

```
:- set_prolog_flag(editor,pce_emacs).
```

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 12

Directories for Propositional logic

Create the following directories:

- Propositional/Basis
- Propositional/Evaluations
- Propositional/Tests
- Propositional/Transformations

Programs for truth tables

From the course's web site, download the following files, to copy in the following directories:

- In Propositional/Basis:
external_operators.pl,
internal_operators.pl, translations.pl, and
truth_tables.pl.
- In Propositional/Evaluations,
truth_table.pl
- In Propositional/Tests,
truth_table_tests.pl

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 13

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 14

Computing truth tables

In Propositional/Tests, run swipl, load
truth_table_tests.pl and test queries t1 to t11.

?- [truth_table_tests].

?- t1.

p	->	r	p	r	value
			t	t	t
			t	f	f
			f	t	t
			f	f	t

Yes

Lecture notes 3.0, COMP 2411, session 1, 2004 – p. 15