Database manipulation

A Prolog program can be viewed as a database: a specification of a set of relations given as facts and rules.

We might encounter situations where we'd like to update the database, either by adding facts or rules, or by deleting facts or rules.

The built-in predicates `assert` and `retract` allow us to do exactly that.

Database manipulation

A Prolog program can be viewed as a database: a specification of a set of relations given as facts and rules.

We might encounter situations where we'd like to update the database, either by adding facts or rules, or by deleting facts or rules.

The built-in predicates `assert` and `retract` allow us to do exactly that.

Make no mistake however -- these predicates take us way outside of Pure Prolog.

Database manipulation

```
?- assert(<some clause>).
```

Adds the clause to the corresponding procedure. Where? In SWI-Prolog, the clause is added to the end of the procedure.

```
?- asserta(<some clause>).
```

Adds the clause to the beginning of the corresponding procedure.

```
?- assertz(<some clause>).
```

Adds the clause to the end of the corresponding procedure.

Database manipulation

```
?- retract(<some clause>).
```

Finds the first clause in the database that matches (i.e., unifies with) `<some clause>` and removes it from the database. (How that's implemented may vary from one Prolog implementation to another.)

Database manipulation

`assert` and `retract` work only on dynamic predicates. So far, everything you've created is a static predicate. Built-in predicates are also static. If you want a predicate to be modifiable, you need to declare it as such:

```
:- dynamic <pred_name>/<arity>, … ,
           <pred_name>/<arity>.
```

Here's an example:

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r66,r67).
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r66,r67).
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).


?- retract(connects_to(r66,r67)).
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r66,r67).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).



?- retract(connects_to(r66,r67)).

true
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r66,r67).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).



?- assert(connects_to(r66,r67)).
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r66,r67).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).
connects_to(r66,r67).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).


?- assert(connects_to(r66,r67)).

true
```

Database manipulation

Things to keep in mind:

- `assert` and `retract` are computationally expensive (`assert` compiles its clause; `retract` has to decompile the program to unify its clause before retracting)

Database manipulation

Things to keep in mind:

- `assert` and `retract` are computationally expensive (`assert` compiles its clause; `retract` has to decompile the program to unify its clause before retracting)

- "Excessive and careless use of these facilities cannot be recommended as good programming style….relations that hold at some point will not be true at some other time. At different times the same questions receive different answers. The resulting behaviour of the program may become difficult to understand, difficult to explain and to trust." [from *Prolog Programming for Artificial Intelligence* by Ivan Bratko]

Database manipulation

Things to keep in mind:

- "The predicates `assert` and `retract` introduce to Prolog the possibility of programming with side effects. Code depending on side effects for its successful execution is hard to read, hard to debug, and hard to reason about formally. Hence these predicates are somewhat controversial, and using them is sometimes a result of intellectual laziness or incompetence. They should be used as little as possible when programming." [from your textbook, *The Art of Prolog*]

Database manipulation

Things to keep in mind:

- Sometimes you gotta do what you gotta do.

Database manipulation

Things to keep in mind:

- Sometimes you gotta do what you gotta do.
- Read Chapter 12.2 and
  the SWI-Prolog manual (4.13) for more details.