Self-inspecting and self-modifying programs

- Tools in Prolog
- Meta-interpreters (short intro)
- Modifying program while it runs

Metaprogramming: treating programs as data

Homogeneous format, program ≈ data

Using infix op. for:
`':-'(p(X=,q(X))`

- Can you tell difference between

  fact **`p(a,b)`** and *term* **`p(a,b)`**?

  clause **`p(X):- q(X)`** and *term* **`p(X):- q(X)`**?

Self-inspection by predicate clause/2.

Works as if any clause *head*:- *body* represented dually by fact

`clause(`*head*, *body***`).`**

NB: works only when predicates are declared to be dynamic:

```
:- dynamic father/2, grandfather/2.
```

Vanilla: A (meta-)interpreter for Prolog in Prolog

```
solve(true):- !.
solve((A,B)):- !, solve(A), solve(B).
solve(A):- clause(A,B), solve(B).
```

```
write('Try: '),
write((A:- B))
```

```
write('Success: '),
write((A:- B))
```

Querying it:

```
?- father(X,Y).
X = john, Y = mary ? ;
X = john, Y = karen
?- solve(father(X,Y)).
X = john, Y = mary ? ;
X = john, Y = karen
```

Why on earth...???

You can modify it, e.g.,

- add test prints 🟩
- change order of evaluation ~

A more advanced example:
Relaxation by taxonomy,
  if subgoal empty, step up in taxonomy
  dog? empty! animal? cat? nonempty :)

Modifying the program while it runs

Add new *last* clause: **`assertz(`***clause***`)`**

Add new *first* clause: **`asserta(`***clause***`)`**

Delete *first clause unifying* pattern: **`retract(`***clause***`)`**

```
?- father(X,Y).
X = peter, Y = karen ?;  // no
?- asserta(father(john,mary)),assertz(father(john,paul)).
yes
?- father(X,Y).
X = john, Y = mary ?;
X = peter, Y = karen ?;
X = john, Y = paul ?; // no
?- \+ (retract(father(john,X)), fail). // yes
?- father(X,Y).
X = peter, Y = karen ?;  // no
```

Applications for AI

- Metainterpreters, for modifying execution strategy, adding "meta-rules" (a la expert systems)
- Defining backward/forward chaining with assert for "modifying the fact base"
  - (I have never tried this; a good exercise ...)

- Later we introduce Constraint Handling Rules in which these things can be explained in more clean way.