**Self Reflection**

**Prolog Database**
- The working environment of Prolog, containing all loaded Prolog programs is called: the 'database'
- The database can be manipulated by the programs themselves
- Compare: Pascal program that modifies itself during execution

**Prolog ‘Database’**
- add new clauses
- remove clauses

**Prolog ‘Database’**
- `assertz`: add to the end of a definition
- Example:

```
parent(jim, bob).
pred(X,Y) :-
  parent(X,Y).
pred(X,Z),
pred(Z,Y).
```

**Asserting Clauses**

```
Database
| collect_data(stop),
| collect_data(_, _) :-
| write("Next item: "),
| read(X),
| assert(X),
| collect_data(X).

input_data :-
  collect_data(start).

name(peter),
  age(35),
  stop.
```

**Database Manipulation**
- **Asserting (new) clauses:**
  - `assert(C)`: position C unspecified
  - `asserta(C)`: at the beginning of the definition of the predicate
  - `assertz(C)`: at the end of the definition of the predicate
- **Deleting clauses:**
  - `retract(C)`: remove clause matching with C (top to bottom order)
Retracting Clauses

Retract: remove from the beginning of the definition

?- retract(parent(X,Y)),
X = jim
Y = bob
yes

Prolog Database

?- dynamic parent/2.
  parent(jim, bob).
  parent(bob, ann).
  parent(john, pete).
  parent(pete, linda).

?- retract_all_facts(parent(X,Y)).
yes

Art of Prolog Programming

- Write correct programs: first think about how to represent the problem in Prolog (postpone all other issues to a later stage)
- declarative design correct
- termination

- Readability: structure, layout and documentation
- Modifiability
- Efficiency: add cuts, pay attention to order, choose efficient representation

Layout

merge([], L, L).
merge([H|T1], [], L).
merge([H|T1], [H2|T2], [H|T3]) :-
  H1 < H2, L2,
  merge(T1, [H2|T2], [H3|T3]) :-
  merge(L1, T2, T3).
append([], L, L).
append([H|T], U, [H|W]) :-
  append(T, U, W).

Bad Example 🎭 😞

m(L1,L2,L3):-
L1=[], L, L3=L2,
L2=[], L, L3=L1,
L1=[X|T1], L2=[Y|T2],
(X<Y, !, Z=X, m(T1,L2,T3);
Z=Y, m(L1,T2,T3)),
L3=[Z|T3].
a([],X,X).
a([X|Y],V,[X|U]):-a(Y,V,U).

Applications of Prolog

- Artificial Intelligence:
  - natural language processing
  - symbolic reasoning systems, like expert systems and qualitative simulation
  - planning systems

- Bioinformatics:
  - recognition of DNA fragments
  - recognition of 3-D structure of proteins

- Specification languages: implementation of interpreters/compilers
Why is Prolog so Good for Solving AI Problems?
- Simple to define representations, e.g.: description(car, isa(vehicle), [wheels(4), maxspeed(200)]).
- Easy to manipulate representations:
  ?- description(Object, isa(Object2), Property_list)

Prolog and Bioinformatics
?- Find proteins with 2 x 2 linked circles

Qualitative Simulation
- Simulation of behaviour of system (e.g. circuit), using Prolog
- Example: multiplier-added

Final Words
- Prolog also used for Internet-like applications:
  - as Prolog Web server
  - as language for client-side programs

```
/* behaviour of */
/* multiplier */
output(O,M) :-
  multiplier(M), input1(M, 11), input2(M, 12), O is 11 * 12.
```