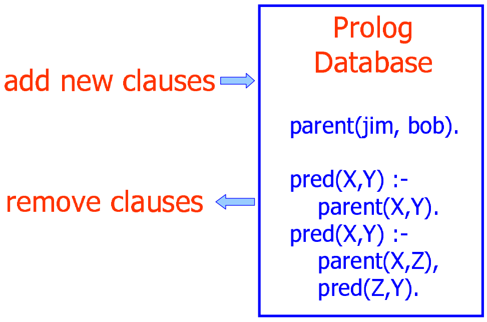
Self Reflection

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

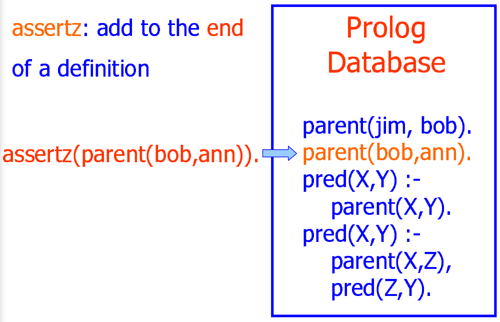
remove clauses

Prolog Database

```
parent(jim, bob).

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

Prolog 'Database'



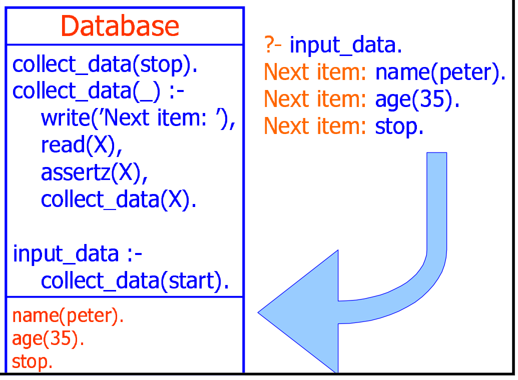
assertz: add to the end of a definition

assertz(parent(bob,ann)).

Prolog Database

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

Asserting Clauses



Database

```
collect_data(stop).
collect_data(_) :-
    write('Next item: '),
    read(X),
    assertz(X),
    collect_data(X).

input_data :-
    collect_data(start).
```

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

```
?- input_data.
Next item: name(peter).
Next item: age(35).
Next item: 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 of 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).
```

Retracting Clauses

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

Prolog Database

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

```
retract_all_facts(X) :-
    retract(X),
    fail.
retract_all_facts(_).
```

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

```
/***************************************/
/* List manipulation programs.         */
/* merge(U, V, W): merges two ordered lists   */
/***************************************/

merge([], L, L) :- !.
merge(L, [], L).
merge([H1|T1], [H2|T2],[H3|T3]) :-
    H1 < H2, !,
    merge(T1, [H2|T2], T3).
merge(L1, [H1|T2], [H2|T2]) :-
    merge(L1, T2, T3).

append([], L, L).
append([H|T], U, [H|W]) :-
    append(T, U, W).
```

blank

no empty line

indentation

empty line

Bad Example

```
m(L1,L2,L3):-
L1=[], !, L3=L2,
L2=[], !, 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?

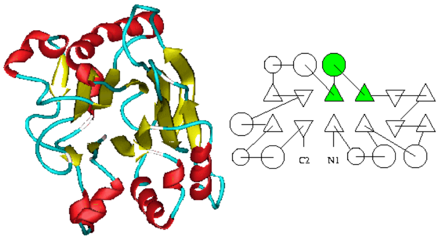
- Simple to *define* representations, e.g.:

```
description(car,
    isa(vehicle),
    [wheels(4),
     maxspeed(200)]).
```

- Easy to *manipulate* representations:

```
?- description(Object,
               isa(Object2),
               Property_list)
```

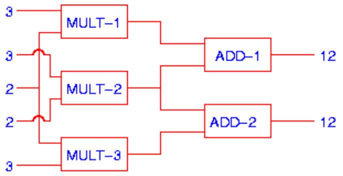
Prolog and Bioinformatics



?- Find proteins with 2 x 2 linked circles

Qualitative Simulation

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

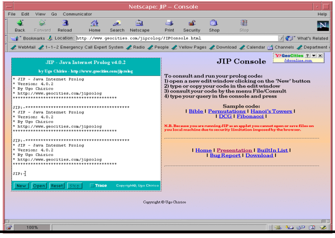


```
/* behaviour of */
/* multiplier    */

output(O,M) :-
    multiplier(M),
    input1(M, I1),
    input2(M, I2),
    O is I1 * I2.
```

Final Words

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


Netscape: JIP – Console
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://www.geocities.com/jiprolog/JIPConsole.html
What's Related
WebMail 1-1-2 Emergency Call Expert System Radio People Yellow Pages Download Calendar Channels Department
GeoCities Jobsonline.com
New Open Reset Stop Trace
100%

JIP - Java Internet Prolog v4.0.2

by Ugo Chirico - http://www.geocities.com/jiprolog

```
* JIP - Java Internet Prolog
* Version: 4.0.2
* By Ugo Chirico
* http://www.geocities.com/jiprolog
******************************************

JIP:-******************************************
* JIP - Java Internet Prolog
* Version: 4.0.2
* By Ugo Chirico
* http://www.geocities.com/jiprolog
******************************************

JIP:-******************************************
* JIP - Java Internet Prolog
* Version: 4.0.2
* By Ugo Chirico
* http://www.geocities.com/jiprolog
******************************************

JIP:-
```

Copyright©, Ugo Chirico

JIP Console

To consult and run your prolog code:
1) open a new edit window clicking on the "New" button
2) type or copy your code in the edit window
3) consult your code by the menu File/Consult
4) type your query in the console and press

Sample code:
| Bible | Permutations | Hanoi's Towers |
| DCG | Fibonacci |

N.B. Because you are running JIP as an applet you cannot open or save files on you local machine due to security limitation imposed by the browser.

| Home | Presentation | BuiltIn List |
| Bug Report | Download |

Copyright © Ugo Chirico