Introduction to Prolog

Basanta Joshi, PhD
Asst. Prof., Depart of Electronics and Computer Engineering
Program Coordinator, MSc in Information and Communication Engineering
Member, Laboratory for ICT Research and Development (LICT)
Institute of Engineering
basanta@ioe.edu.np
http://www.basantajoshi.com.np

LOG: PROgramming in LOGic

loped in 1970 in France

arative Language unlike LISP, PASCAL, C, . . . v
procedural languages

ssignment statements

No got

No if-then-else structures

No loops

The user describes the problem in the form
of facts and rules and PROLOG applies

og is a combination of the following key ideas:

then rules with variables

elational databases (w/ terms for data structurin

ackward Chaining to try to prove goals

nification to match goals to rule conclusions.

acktracking to try all possibilities

e are several free PROLOG Compilers available

/I Prolog is a non-commercial product. This is

PCs in Memorial.

nzi! Prolog is available on a 90 day free trial offe

sual Prolog has a student version

rawberry Prolog is a nice light version

OLOG works with facts, relations and rules.

act is a unit of information which is assumed to
ue.

elation combines two or more facts.

ule is a conditional assertion of a fact.

scription of Ann's and Sue's world

Description	PROLOG Version
oll is a toy.	toy(doll).
opy is a toy.	toy(snoopy).
plays with Snoopy.	plays(ann,snoopy).
likes everything Ann likes.	likes(sue,Y) :- likes(ann,Y);
likes the toys she plays with	likes(ann,X) :- toy(X),plays(a

Every PROLO
is a claus

A constant is lower case

A variable is upper c

n the description of Ann's and Sue's world, the
stion might be asked: "What does Sue like?"

PROLOG: |?-likes(sue,X).

X = snoopy

PROCESS: Prolog tries to find a value for all variables in a query so as to make the query true.

Sue likes X if Ann likes X

Ann likes X if X is a toy and Ann plays with X

Snoopy is a toy and Ann plays with Snoopy

So, Sue likes Snoopy

nter a PROLOG program:

|?-consult(user).

<rules/facts/relations>

end-of-file.

or CRTL - D

Alternative: Save the program in a file and use |?-consult(<filename>).

uery returns a single answer, if you want to find ernative answers, respond to a PROLOG result th a “**;**” and PROLOG will seek another way to ake the query true.

ll report the new answer or “**no**” if there are not her solutions.

Query Forms

```
|?- likes(sue,snoopy).
Yes
|?-likes(sue,ann).
no
```

```
|?- toy(X),likes(sue,X)
```

Prolog Programming

DAL: Build a travel database and write PROLOG
uery routines

gin by defining the structure of any relationships

travel(carrier,origin,destination,ty

travel(amtrak, new-york, boston,train).
travel(nj-transit, new-york, boston,train).
travel(amtrak, boston, portland,train).
travel(greyhound, boston, portland,bus).
travel(amtrak, new-york, washington,train)

olog program consists of rules and facts

L: Construct a rule that will identify competitors

is the definition of a competitor
in this small travel world?

Two carriers are competitors if the
travel between the same two cities

variables

if

and

anonymous va

```
competitor(Carrier1,Carrier2) :-
    travel(Carrier1, CityA, CityB, _),
```

DAL: Develop a rule that will determine if it is ossible to travel between two cities

We can travel between A and B if there is a carrier which starts at A and ends at B.

```
can-travel(CityA, CityB) :-
    travel(_, CityA, CityB, _).
```

Don't care about the carrier name

or type

n this travel base and the can-travel rule what is the
onse to the query:

n-travel(new-york,portland).

t correct? If not, what is the problem?

w should a more general can-travel rule be expressed
lish?

travel(amtrak, new-york, boston,tr
travel(nj-transit, new-york, boston,
travel(amtrak, boston, portland,tr
travel(greyhound, boston, portland
travel(amtrak, new-york, washington

n travel from A to B if we can travel
A to C and then from C to B for any
mber of intermediate cities, C.

ore general can-travel rule involves two versions
he can-travel relation:

ROLOG will try this first

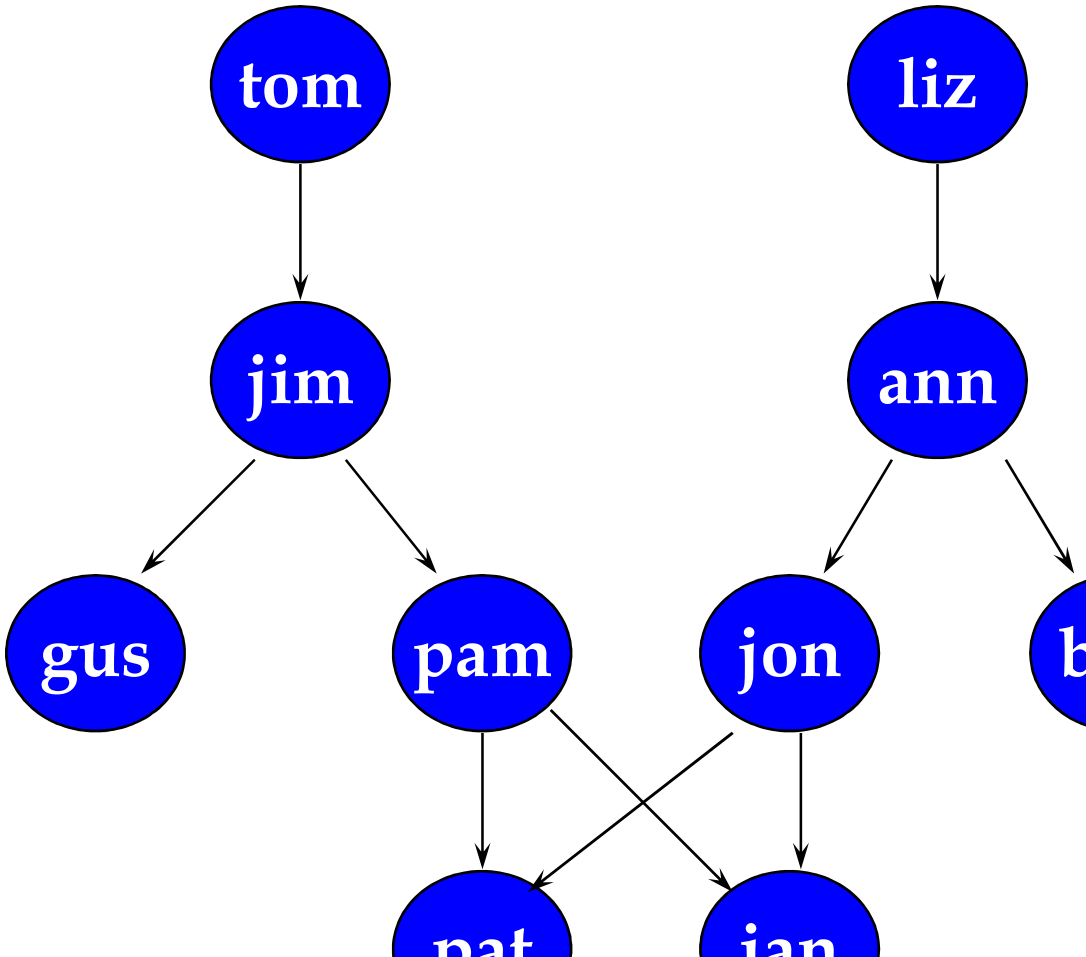
```
can-travel(CityA, CityB) :-
    travel(_, CityA, CityB, _).
```

n this rule if the first fails

```
can-travel(CityA, CityB) :-
    travel(_, CityA, CityC, _),
    can-travel(CityC, CityB).
```

n a family tree – represent it in Prolog:

```
parent(tom, jim).
parent(jim, gus).
parent(jim, pam).
parent(pam, pat).
parent(pam, jan).
parent(liz, ann).
parent(ann, jon).
parent(ann, bob).
parent(jon, pat).
```



can ask several questions about this database:

am a parent of pat?

```
arent(pam, pat).
    yes
```

Is pam a parent of a

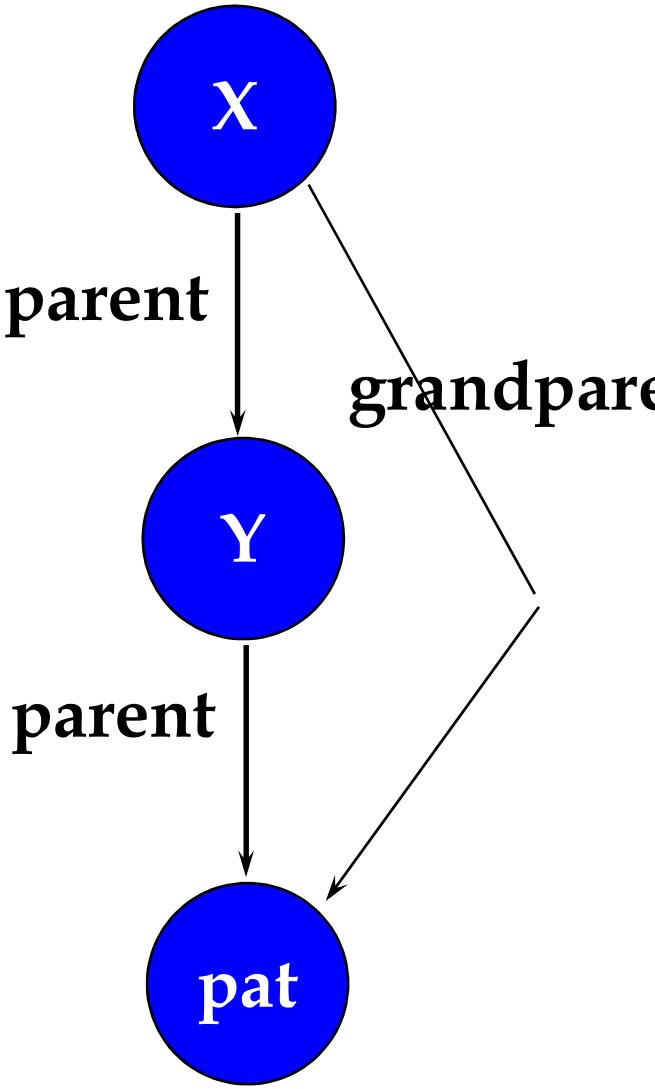
```
?- parent(pam, X)
    X = pat     ;
    X =  jan    ;
        no
```

Do gus and jan share the same parent?

```
?- parent(X, gus), parent(X, jan)
        X = jim
```

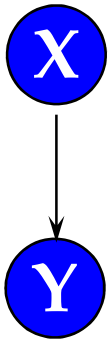
etermine a grandparent:

?- parent(X,Y),parent(Y,pat).
X = jim
Y = pam



For parents

decessor(X, Y) :- parent(X, Y).

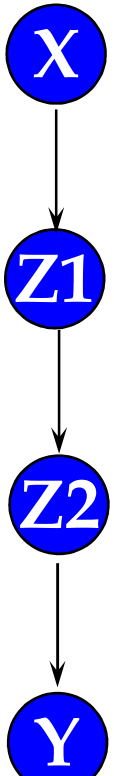


For grandparents

predecessor(X, Y) :-
arent(X, Z1), parent(Z1, Y).

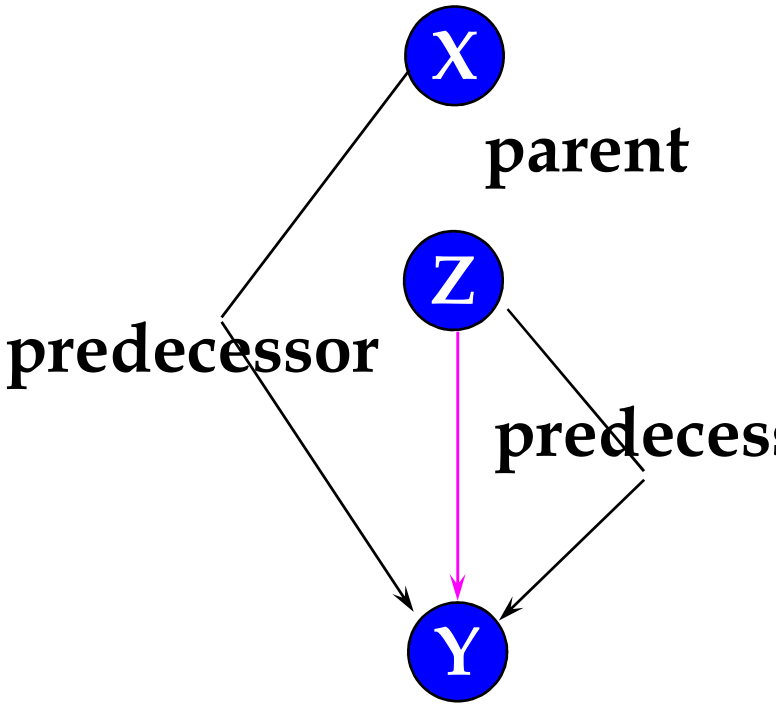
For great grandparents

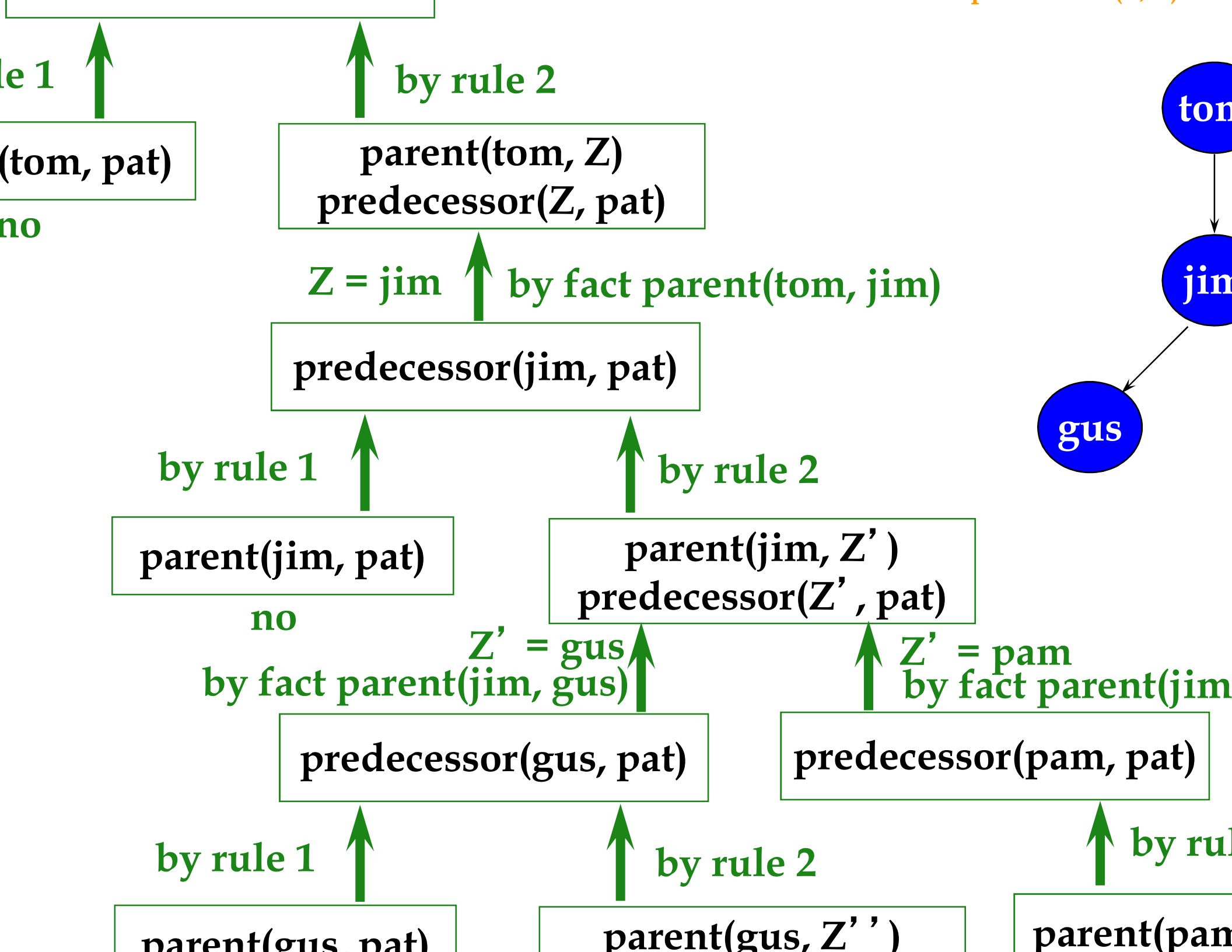
redecessor(X, Y) :- parent(X, Z1),
parent(Z1, Z2),
parent(Z2, Y).



```
redecessor(X, Y) :- parent(X, Y).

edecessor(X, Y) :- parent(X, Z),
                   predecessor(Z, Y).
```




le 1
(tom, pat)
no
by rule 2
parent(tom, Z)
predecessor(Z, pat)
Z = jim
by fact parent(tom, jim)
predecessor(jim, pat)
by rule 1
parent(jim, pat)
no
by rule 2
parent(jim, Z')
predecessor(Z' , pat)
Z' = gus
by fact parent(jim, gus)
Z' = pam
by fact parent(jim
predecessor(gus, pat)
predecessor(pam, pat)
by rule 1
by rule 2
by rul
parent(gus, pat)
parent(gus, Z' ')
parent(pam
tom
jim
gus

g searches for clauses (rules & facts) from top t
tom

g attempts goals from left to right

ood advice

Put simplest clauses first

Put simplest goals on the left

though logically correct, Prolog can’t handle the
ollowing

```
predecessor(X, Y) :- predecessor(Z, Y), paren
Z).
predecessor(X, Y) :- parent(X, Y).
```

Prolog I/O

ere are two I/O commands in PROLOG

te(X). - writes a term X to the current output

d(X). - reads a term X from the current input.

EXAMPLE: given color(blue) color(red) the

```
|?- color(X), write(X), nl.
blue   ;
```

```
|?- read(X), write(X).
test.
```

s useful to provide a prompt for a read operation

Define a rule input(X)

Write the prompt

Read the value

```
input(X) :-
    write( '>' ),
    read(X).
```

Running Prolog

n SWI and a Prolog window opens

t a session

r the code

er a query

```
SWI-Prolog (version 3.1.0)

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(user).
|: parent(tom, jim).
|: parent(jim, gus).
|: parent(jim, pam).
|: parent(pam, pat).
|: parent(pam, jan).
|: parent(liz, ann).
|: parent(ann, jon).
|: parent(ann, bob).
|: parent(jon, pat).
|: parent(jon, jan).
|:
user compiled, 89.41 sec, 952 bytes.

Yes
?- parent(X, jim).

X = tom ;

No
?-
```

save a program and run it at a later me

reate the program as a set of rules and facts in a SCII file

ave it with a .pl extension

art SWI Prolog and enter

consult(<filename>).

he file will be read into Prolog and you are now ady to enter queries

eate a text file with the following set of rules and
cts:

```
arent(tom, jim).
arent(jim, gus).
arent(jim, pam).
arent(pam, pat).
arent(pam, jan).
arent(liz, ann).
arent(ann, jon).
arent(ann, bob).
arent(jon, pat).
```

New Facts

```
female(pam).
female(pat).
female(liz).
female(ann).
female(jan).
male(tom).
male(jim).
male(gus).
male(jon).
male(bob).
```

Mother Rule:

X is the mother of
X is a parent of Y a
X is a female.

```
mother(X,Y):-
parent(X,Y),female
```

SWI and consult your program file:

e a query

```
SWI-Prolog (version 4.0.9)
Copyright (c) 1990-2000 University of Amsterdam.
Copy policy: GPL-2 (see www.gnu.org)

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(\family).
% \family compiled 0.00 sec, 2,412 bytes

Yes
?- mother(X, pat).

X = pam

Yes
?-
```

Prolog Examples

g the family tree data base consider the rule:

```
ster(X, Y) :- parent(Z, X), parent(Z, Y),
male(X).
```

nfortunately Prolog succeeds on

```
sister(ann, ann).
```

olution

```
sister(X, Y) :- parent(Z, X), parent(Z, Y), female(X
X\=Y.
```

assignment like statement is in the form:

X is E .

The arithmetic expression **E** is evaluated and the
ariable **X** is instantiated to the result.

```
?- N is 3 *
4.

N = 12

yes
```

```
?- X is 3 +
4/5.

X = 3.8
```

BLEM: Find the factorial of a nonnegative integ

= is a test, not an assignment

```
torial(N,F) :- N=0, F=1.

torial(N,F) :- N>0,
               N1 is N - 1,
               factorial(N1,F1),
               F is N*F1.
```

ENGLISH: F is t
factorial of N if eit
N=0 and F = 1 or N
and F=N*F1 whe
F1 is the factorial o
1.

e a rule to find the maximum of two numbers X
Y

Relation: **max(X,Y,Max)**

sh: X is the max if X > Y

Max(X,Y,X) :- X > Y

sh: Y is the max if Y >= X

Max(X,Y,Y) :- Y >= X

oblem: Find the square of a number

```
quares :-write( 'Next item please: '),
read(X),
process(X).

cess(stop) :- !.

process(N) :-C is N * N,
e( 'Square of '),write(N),write( ' is '),
write(C), nl,
dosquares.
```

```
?- dosquares.
Next item please: 5.
Square of 5 is 25.
Next item please: 12
Square of 12 is 144
Next item please:
stop
yes
```