Rules

John likes all people

Could list all people

likes(john,alfred).

likes(john,bertrand).

likes(john,charles).

likes(john,david).

.

.

.

Rule is more Compact

John likes any object provided it is a person

Rule Examples

Rules state Dependence

I use an umbrella

If

there is rain

Rules Define

X is a bird

If

X is an animal

and

X has feathers

Formulating Rules

John likes anyone who likes wine

John likes any something if it likes wine

John likes X if X likes wine

Rule Syntax

```
likes(john,X) :- likes(X,wine).
```

head **rule delimitor** **body**

Variable Scope

instantiated here

checked here

returned here

```
likes(john,X) :- likes(X,wine),
                 likes(X,food).
```

X's Within Scope of Rule

Royal Parents

Predicate

The parents of X are Y and Z

Y is the mother

Z is the father

Database

```
male(albert).
male(edward).
female(alice).
female(victoria).
parents(edward,victoria,albert).
parents(alice,victoria,albert).
```

Sisters

X is a sister of Y if:

X is female

X has mother M and father F

Y has mother M and father F

Rule

```
sisters_of(X,Y) :-  female(X),
                    parents(X,M,F),
                    parents(Y,M,F).
```

Scope

```
sister_of(X,Y)  :-
                  female(X),
                  parents(X,M,F),
                  parents(Y,M,F).
```

Scope only within body of rule

Sisters Question

Rule

```
sister_of(X,Y)  :- female(X),
                   parents(X,M,F),
                   parents(Y,M,F).
```

Question

```
sister_of(alice,edward).

sister_of(X,Y)
```

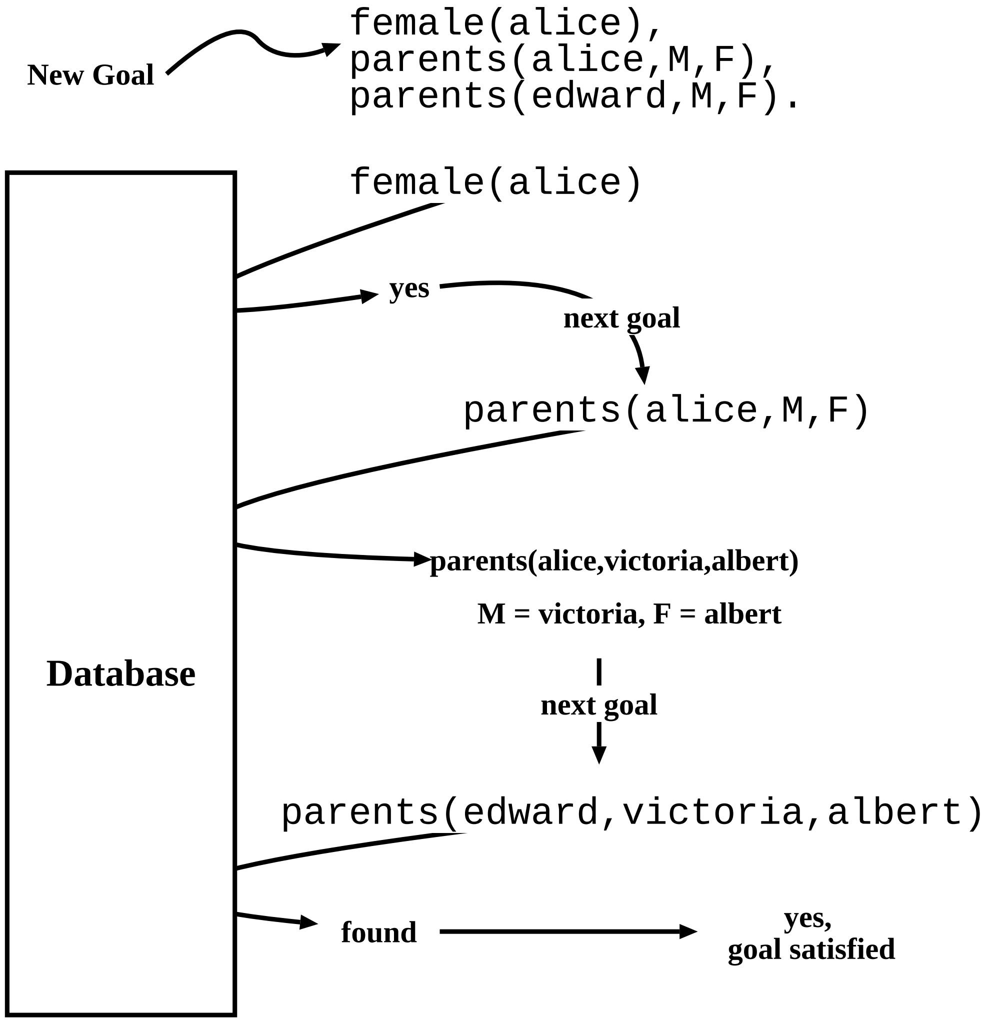
X = alice

Y = edward

New Goal

```
female(alice),
parents(alice,M,F),
parents(edward,M,F).
```

Sisters Question



Who is the Sister?

Rule

```
sister_of(X,Y)  :- female(X),
                   parents(X,M,F),
                   parents(Y,M,F).
```

Question

Different X's

```
sister_of(alice,X).

sister_of(X,Y)
```

X = alice

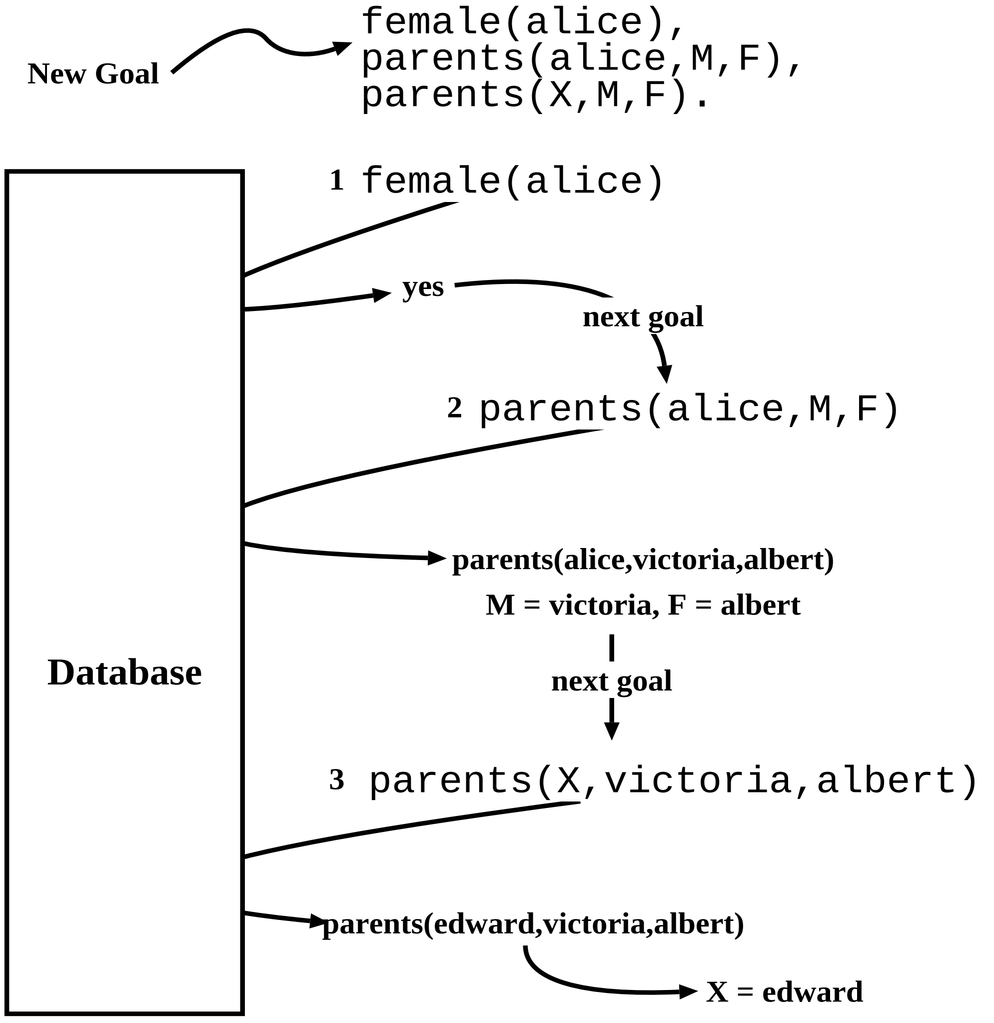
Y = X (toplevel)

uninstantiated variable

New Goal

```
female(alice),
parents(alice,M,F),
parents(X,M,F).
```

Who is the Sister?



Stealing

The Rule

A person may steal something

if

the person is a thief and he likes the thing

Prolog Rule

```
may_steal(P,T) :- thief(P),likes(P,T).
```

Example

```
$ cat thief.pro
thief(john).
likes(mary,food).
likes(mary,wine).
likes(john,X) :- likes(X,wine).
may_steal(X,Y):-thief(X),likes(X,Y).
$
```

Example

```
$ pl
Welcome to SWI-Prolog (Multi-threaded, Version 5.2.6)
Copyright (c) 1990-2003 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

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

?- consult('thief.pro').
% thief.pro compiled 0.00 sec, 1,428 bytes

Yes
?- may_steal(john,X).
```

```
X = mary ;

No
?-
```

Function Trace

```
?- trace.

Yes
[trace] ?- may_steal(john,X).
 + 1  1  Call: may_steal(john,_G27) ?
 + 2  2  Call: thief(john) ?
 + 2  2  Exit: thief(john) ?
 + 3  2  Call: likes(john,_G27) ?
 + 4  3  Call: likes(_G27,wine) ?
 + 4  3  Exit: likes(mary,wine) ?
 + 3  2  Exit: likes(john,mary) ?
 + 1  1  Exit: may_steal(john,mary) ?
X = mary ? ;
```

Redo

```
 + 1  1  Redo: may_steal(john,mary) ?
 + 3  2  Redo: likes(john,mary) ?
 + 4  3  Redo: likes(mary,wine) ?
 + 5  4  Call: likes(wine,wine) ?
 + 5  4  Fail: likes(wine,wine) ?
 + 4  3  Fail: likes(_G27,wine) ?
 + 3  2  Fail: likes(john,_G27) ?
 + 2  2  Redo: thief(john) ?
 + 2  2  Fail: thief(john) ?
 + 1  1  Fail: may_steal(john,_G27) ?

no
[debug] ?- halt.
$
```