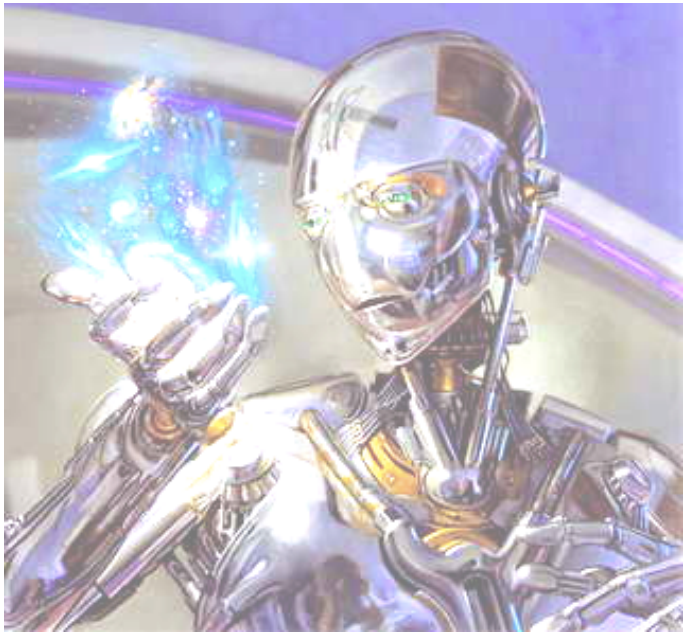
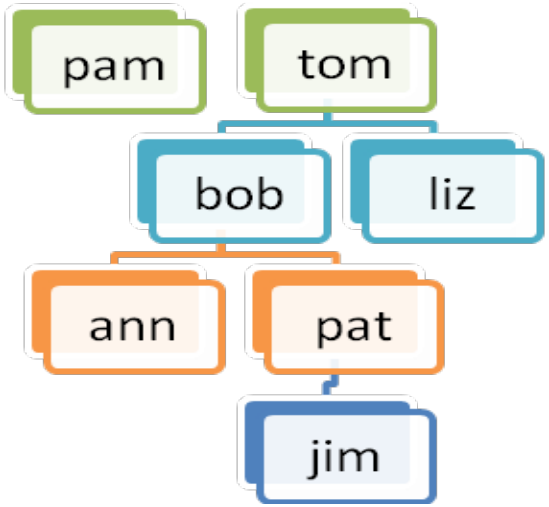


Chapter1 - continued

Recursion

1.3 Recursive rules

- How to express **predecessor** relation in terms of **parent** relation?



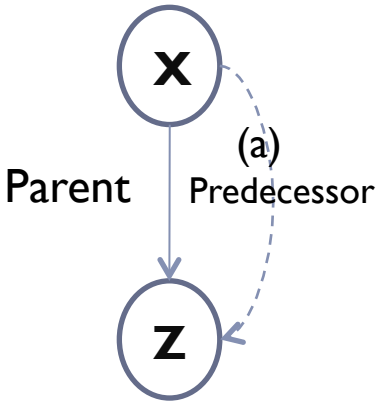
  - It is expressed with two rules:

    1- Direct (immediate) predecessors.

For all **X** and **Z**,

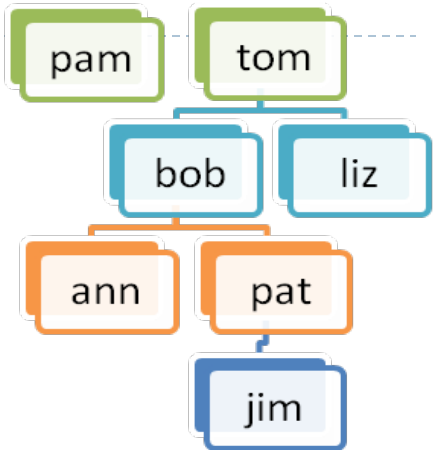
X is a ***predecessor*** of **Z** if

X is a ***parent*** of **Z**




```
predecessor (X, Z):- parent (X, Z)
```

1.3 Recursive rules



2- Indirect predecessors (X is an indirect predecessor of some Z if there is a parentship chain of people between X and Z)

For all **X** and **Z**,

X is a ***predecessor*** of **Z** if

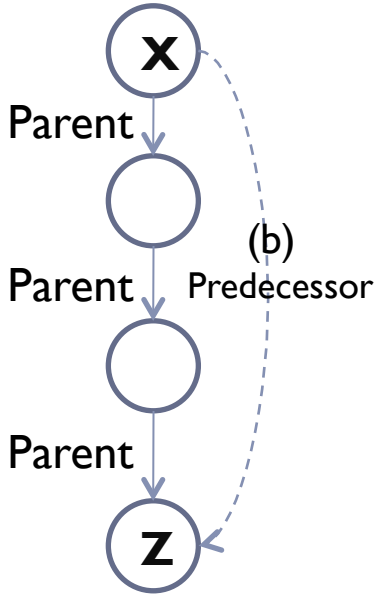
there is **Y** such that

(1) **X** is a ***parent*** of **Y**

(2) **Y** is a ***predecessor*** of **Z**

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

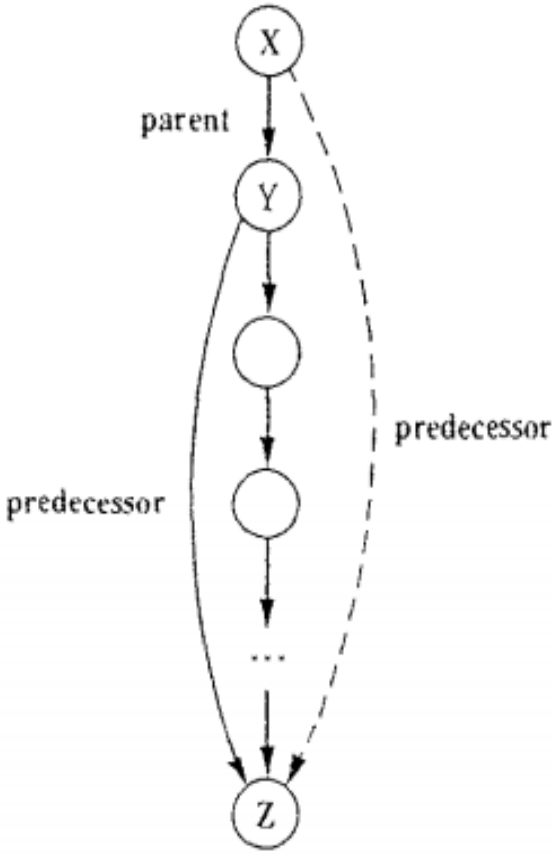
(Recursion)



1.3 Recursive rules

- We have thus constructed a complete program for the "predecessor relation" which consists of 2 rules (procedures) :

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

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



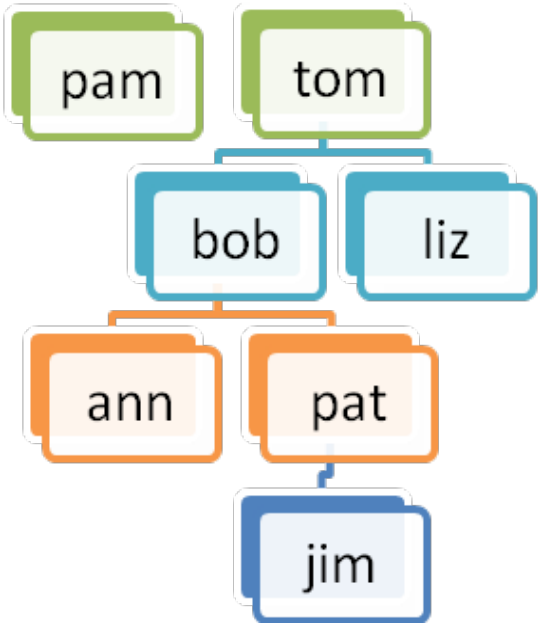
Note that:

- **A recursive clause should always have at least two clauses:**

1) a base clause (the clause that stops the recursion at some point)
2) and one that contains the recursion.

→ If you don't do this, Prolog can spiral off into an unending sequence of useless computations.

Now let us combine all of the relations that we have discussed in the class in a one program that describes the family tree relations of the following situation

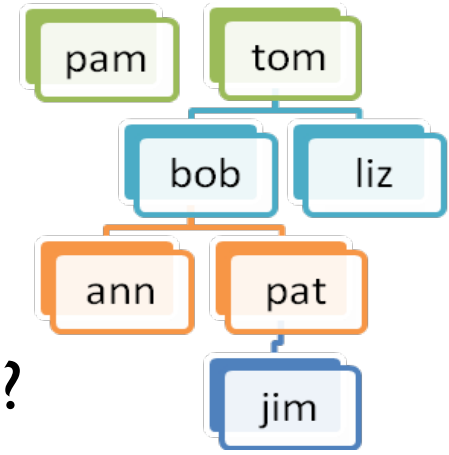


```
parent( pam, bob).      % Pam is a parent of
Bob
parent( tom, bob).
parent( tom, liz).
parent( bob, ann).
parent( bob, pat).
parent( pat, jim).

female( pam).           % Pam is female
male( tom).             % Tom is male
male( bob).
female( liz).
female( ann).
female( pat).
male( jim).

offspring( Y, X)  :-    % Y is an offspring of X if
   parent( X, Y).       % X is a parent of Y

mother( X, Y)  :-       /* X is the mother of Y if
   parent( X, Y),          X is a parent of Y and
   female( X).             X is female */
```

```
grandparent( X, Z)  :-  % X is a grandparent
   parent( X, Y),       % of Z if X is a parent of Y
   parent( Y, Z).       % and Y is a parent of Z

sister( X, Y)  :-       % X is a sister of Y if
   parent( Z, X),
   parent( Z, Y),       % X and Y have the same parent and
   female( X)           % X is female


predecessor( X, Z)  :-  /* Rule pr1: X is a
   parent( X, Z).          predecessor of Z*/

predecessor( X, Z)  :-  % Rule pr2: X is a
     parent( X, Y),     %predecessor of Z
     predecessor( Y, Z).
```

1.3 Recursive rules

- Now we can ask Prolog the following question based on the following family tree
- Is Tom the Predecessor of Pat?




```
?- predecessor ( tom, pat )
```

How will Prolog answer using these 2 procedures ?

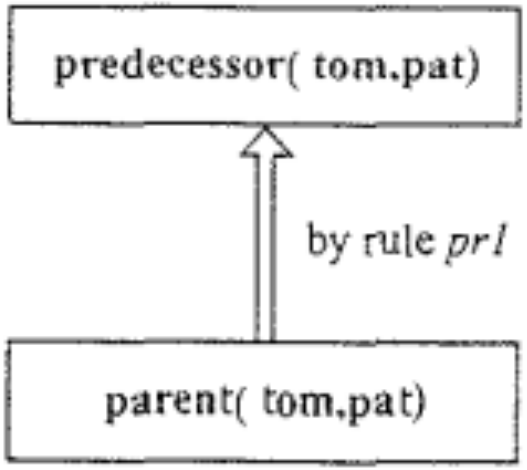
```
predecessor (X, Z):- parent (X, Z) <= Pr1

predecessor(X, Z):- parent (X, Y),predecessor(Y, Z) <= Pr2
```

- Prolog first tries the clause that appears first

```
predecessor (X, Z):- parent (X, Z)
```

```
predecessor (tom, pat):- parent (tom, pat)
```

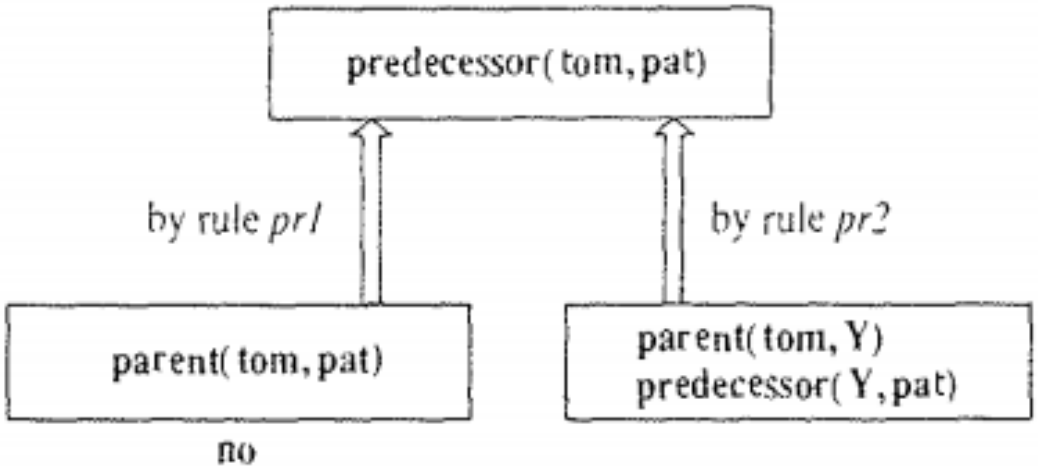
But there is no clause in our program that matches `parent (tom, pat)`, so Prolog backtracks to the original goal to try finding an alternative answer



- The rule Pr2 is thus tried

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

```
predecessor(tom, pat):- parent(tom, Y),predecessor(Y, pat)
```



```
predecessor(tom, pat):- parent(tom, Y),predecessor(Y, pat)
```

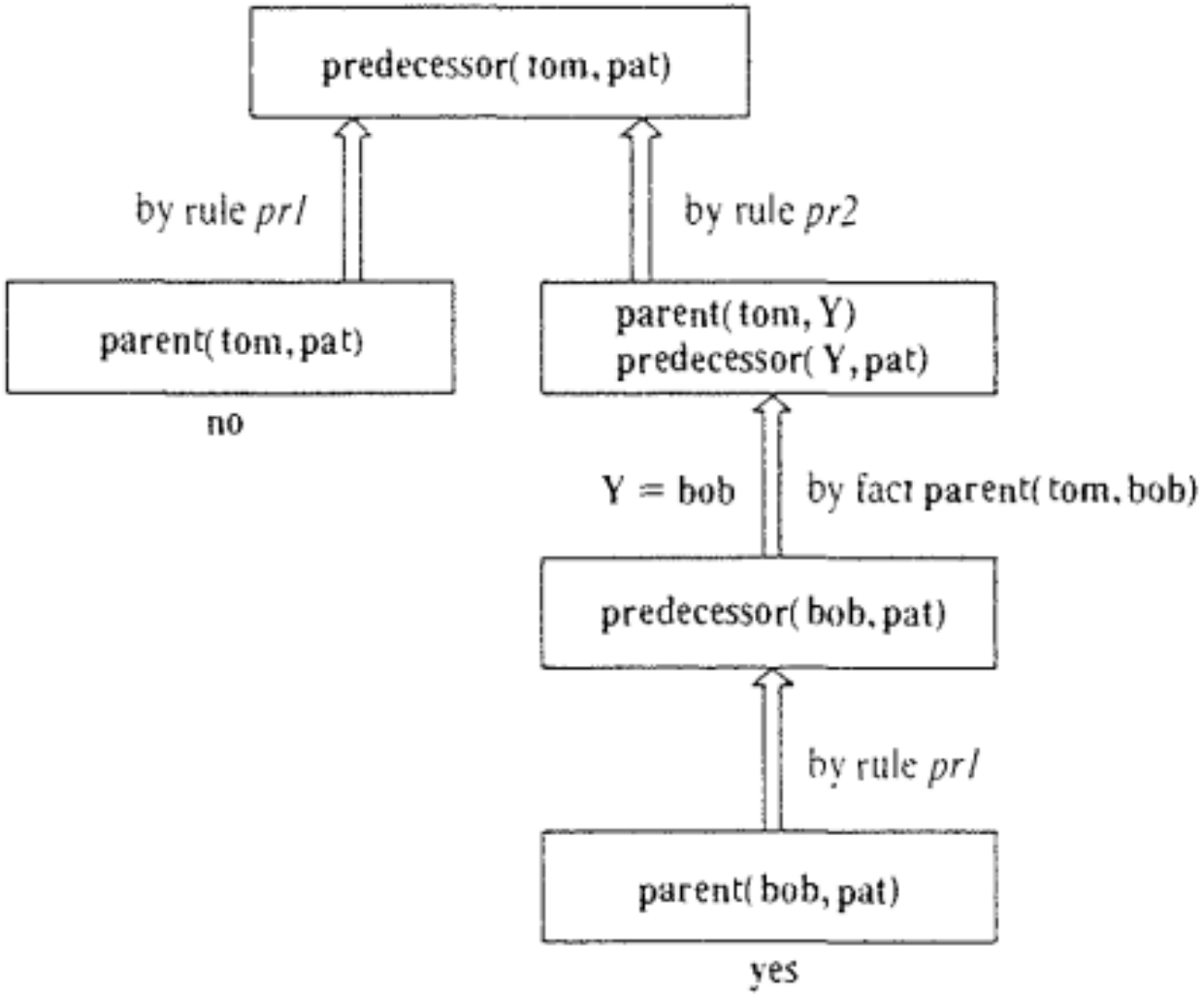
- Prolog is now faced by two goals to satisfy, it starts with the first one `parent(tom, Y)` and try to find a match for `Y`
- So based on the given fact clauses `Y = bob`
- Thus the first goal of the rule is satisfied and the remaining goal is:

```
predecessor(Y, pat) =>  predecessor(bob, pat)
```

Now Prolog will try to satisfy this goal by using the rule of Pr1 again

```
predecessor (X*, Z*):- parent (X*, Z*)
```

```
predecessor (bob, pat):- parent (bob, pat)
```


predecessor(tom, pat)
by rule pr1
by rule pr2
parent(tom, pat)
parent(tom, Y)
predecessor(Y, pat)
no
Y = bob
by fact parent(tom, bob)
predecessor(bob, pat)
by rule pr1
parent(bob, pat)
yes

- This goal is immediately satisfied because it appears as a fact clause in our program

- So Prolog will answer `yes`

Note that:

- Prolog has a specific way of answering queries:
  - Search knowledge base from top to bottom
  - Processes clauses from left to right
  - Backtracking to recover from bad choices