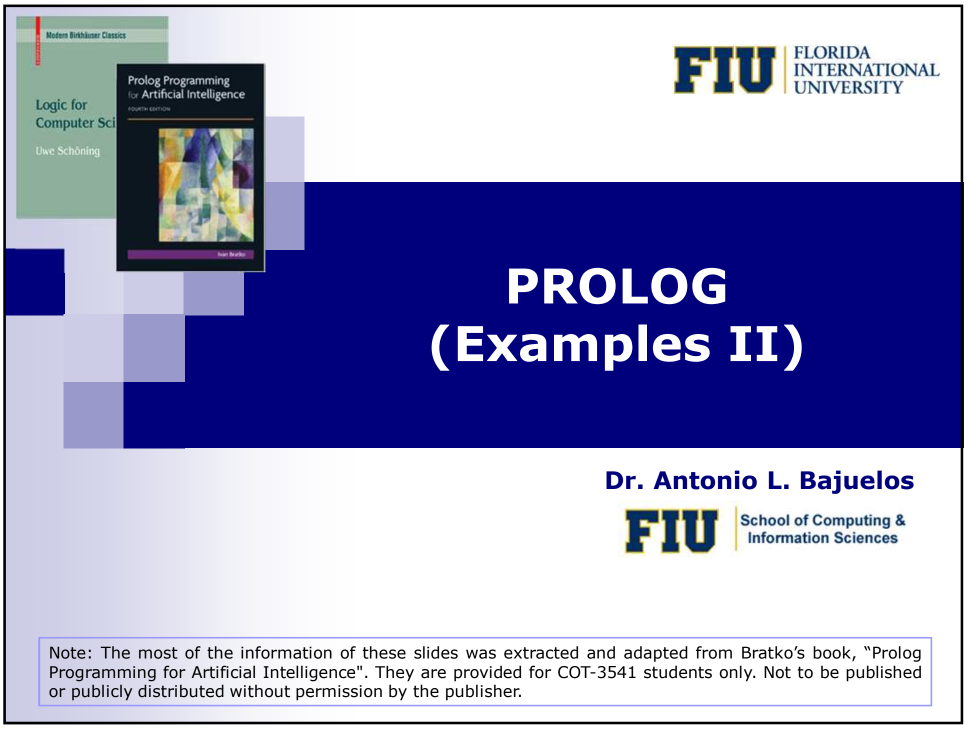


Modern Birkhäuser Classics
Logic for Computer Sci
Uwe Schöning
Prolog Programming for Artificial Intelligence
FOURTH EDITION
Ivan Bratko

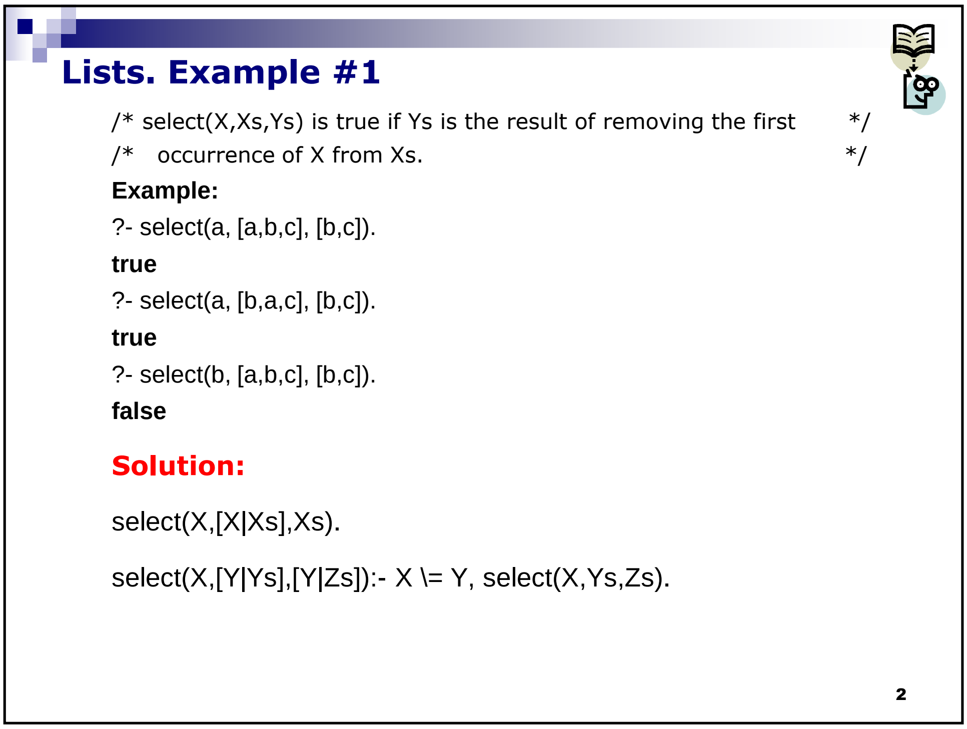
FIU | FLORIDA INTERNATIONAL UNIVERSITY

PROLOG (Examples II)

Dr. Antonio L. Bajuelos

FIU | School of Computing & Information Sciences

Note: The most of the information of these slides was extracted and adapted from Bratko's book, "Prolog Programming for Artificial Intelligence". They are provided for COT-3541 students only. Not to be published or publicly distributed without permission by the publisher.



Lists. Example #1

```
/* select(X,Xs,Ys) is true if Ys is the result of removing the first     */
/*    occurrence of X from Xs.                                            */
```

Example:

?- select(a, [a,b,c], [b,c]).

true

?- select(a, [b,a,c], [b,c]).

true

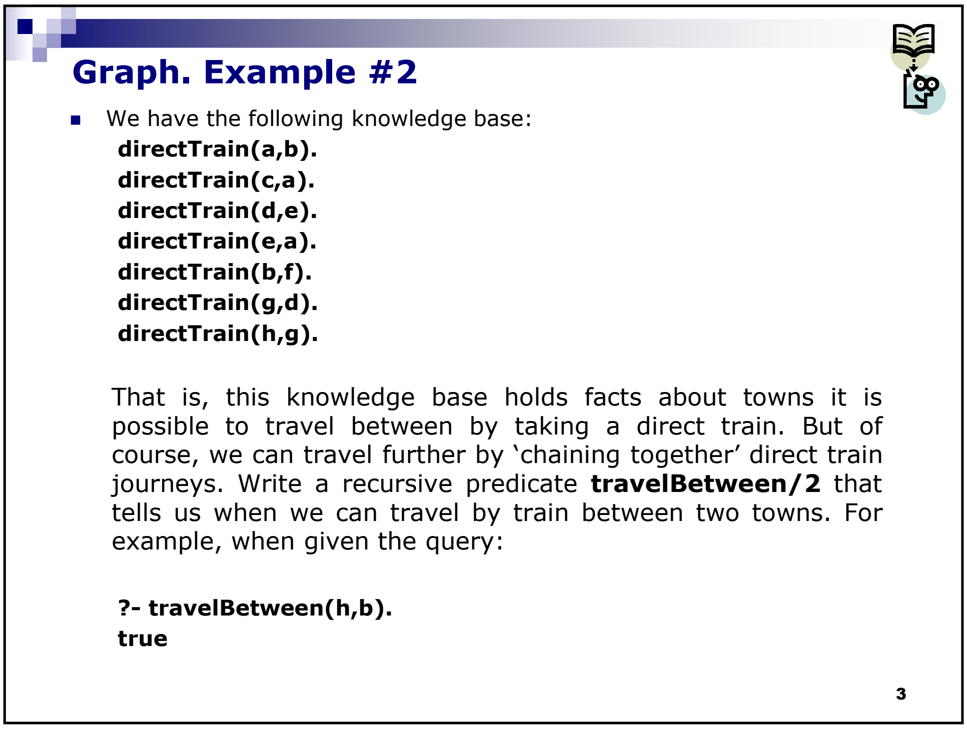
?- select(b, [a,b,c], [b,c]).

false

Solution:

```
select(X,[X|Xs],Xs).

select(X,[Y|Ys],[Y|Zs]):- X \= Y, select(X,Ys,Zs).
```



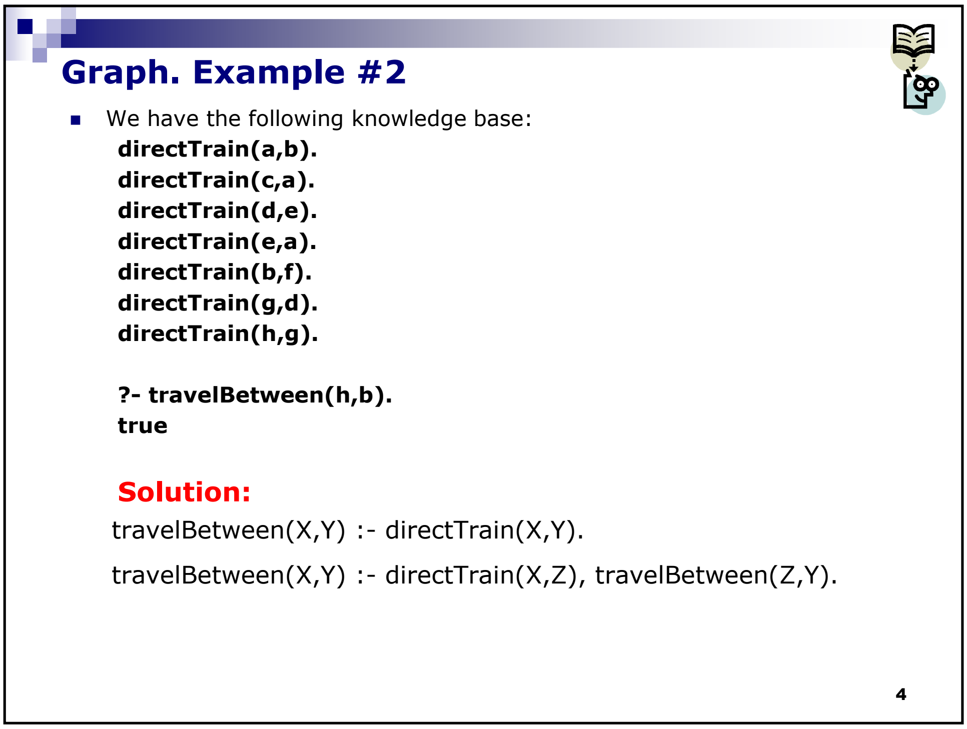
Graph. Example #2

- We have the following knowledge base:

```
directTrain(a,b).
directTrain(c,a).
directTrain(d,e).
directTrain(e,a).
directTrain(b,f).
directTrain(g,d).
directTrain(h,g).
```

That is, this knowledge base holds facts about towns it is possible to travel between by taking a direct train. But of course, we can travel further by 'chaining together' direct train journeys. Write a recursive predicate **travelBetween/2** that tells us when we can travel by train between two towns. For example, when given the query:

```
?- travelBetween(h,b).
true
```



Graph. Example #2

- We have the following knowledge base:

```
directTrain(a,b).
directTrain(c,a).
directTrain(d,e).
directTrain(e,a).
directTrain(b,f).
directTrain(g,d).
directTrain(h,g).

?- travelBetween(h,b).
true
```

Solution:

```
travelBetween(X,Y) :- directTrain(X,Y).
travelBetween(X,Y) :- directTrain(X,Z), travelBetween(Z,Y).
```

Graph. Example #2

- We have the following knowledge base:

```
directTrain(a,b).
directTrain(c,a).
directTrain(d,e).
directTrain(e,a).
directTrain(b,f).
directTrain(g,d).
directTrain(h,g).
```

- It is, furthermore, plausible to assume that whenever it is possible to take a direct train from A to B, it is also possible to take a direct train from B to A. Can you encode this in Prolog? You program should e.g. answer 'yes' to the following query:

```
?- travelBetween(b,h).
true
```

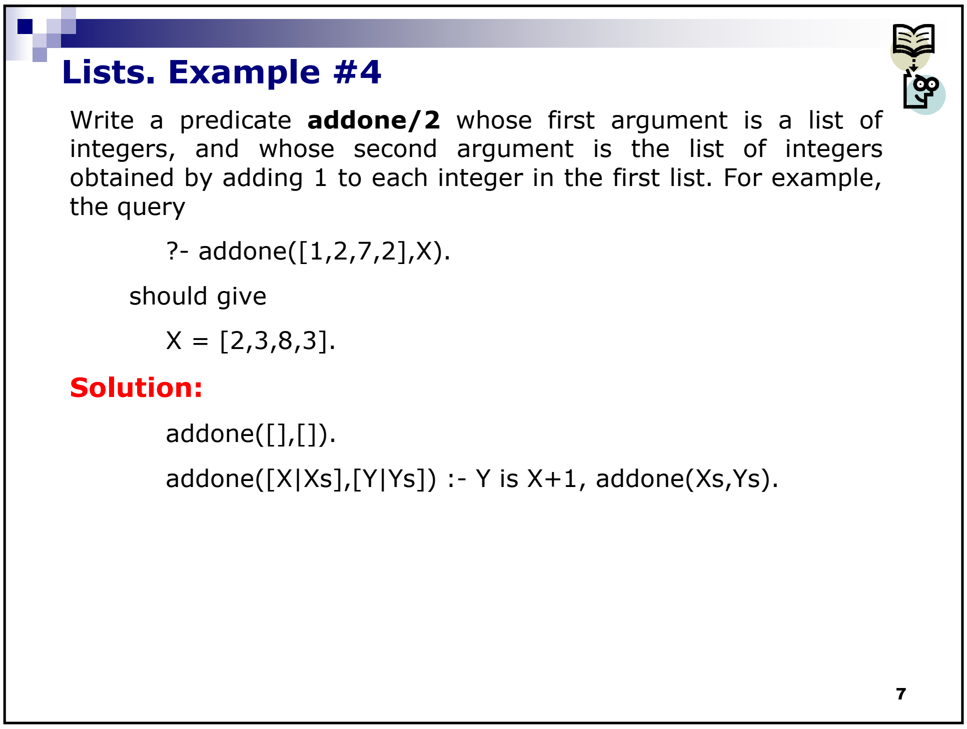
Solution:

Homework!

Lists. Example #3

How does Prolog respond to the following queries?

```
?- [a,b,c,d] = [a,[b,c,d]].
        false
?- [a,b,c,d] = [a|[b,c,d]].
        true
?- [a,b,c,d] = [a,b,[c,d]].
        false
?- [a,b,c,d] = [a,b|[c,d]].
        true
?- [a,b,c,d] = [a,b,c,[d]].
        false
?- [a,b,c,d] = [a,b,c|[d]].
        true
?- [a,b,c,d] = [a,b,c,d,[]].
        false
?- [a,b,c,d] = [a,b,c,d|[]].
        true
?- [ ] = [_].
        false
```



Lists. Example #4

Write a predicate **addone/2** whose first argument is a list of integers, and whose second argument is the list of integers obtained by adding 1 to each integer in the first list. For example, the query

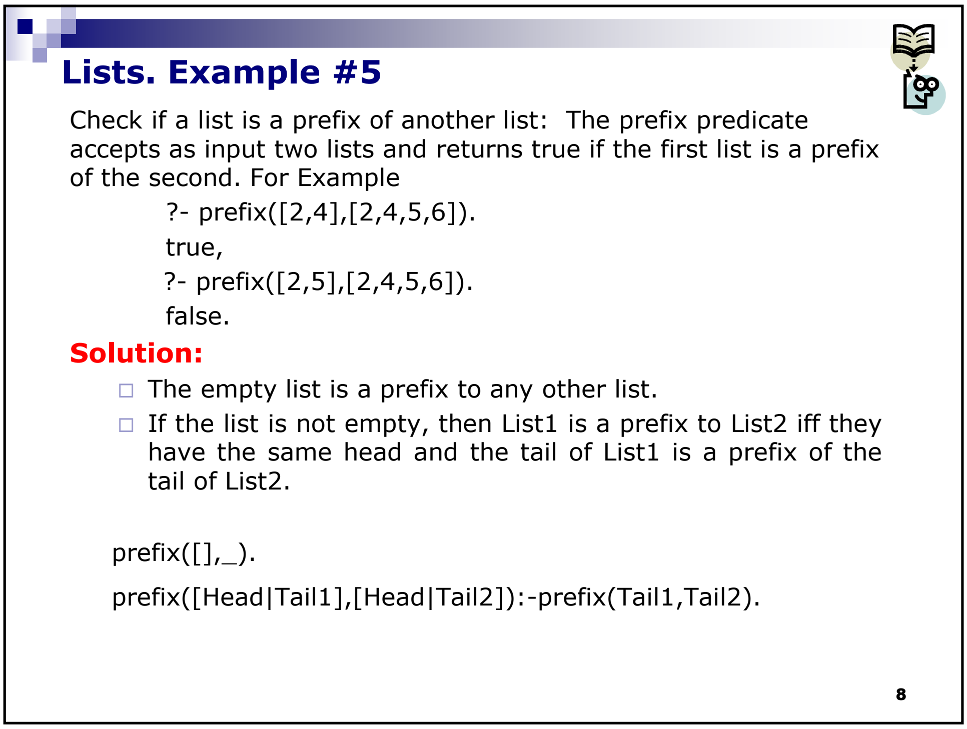
```
?- addone([1,2,7,2],X).
```

should give

```
X = [2,3,8,3].
```

Solution:

```
addone([],[]).
addone([X|Xs],[Y|Ys]) :- Y is X+1, addone(Xs,Ys).
```



Lists. Example #5

Check if a list is a prefix of another list: The prefix predicate accepts as input two lists and returns true if the first list is a prefix of the second. For Example

```
?- prefix([2,4],[2,4,5,6]).
true,
?- prefix([2,5],[2,4,5,6]).
false.
```

Solution:

- □ The empty list is a prefix to any other list.
- □ If the list is not empty, then List1 is a prefix to List2 iff they have the same head and the tail of List1 is a prefix of the tail of List2.

```
prefix([],_).
prefix([Head|Tail1],[Head|Tail2]):-prefix(Tail1,Tail2).
```

Lists. Examples #6 and #7

Write a predicate second(X,List) which checks whether X is the second element of List.

Solution:

```
second(X,[_,X|_]).
```

Write a predicate swap12(List1,List2) which checks whether List1 is identical to List2, except that the first two elements are exchanged.

Solution:

```
swap12([X,Y|T],[Y,X|T]).
```

Cut. Examples #8

Suppose we have the following Prolog database:

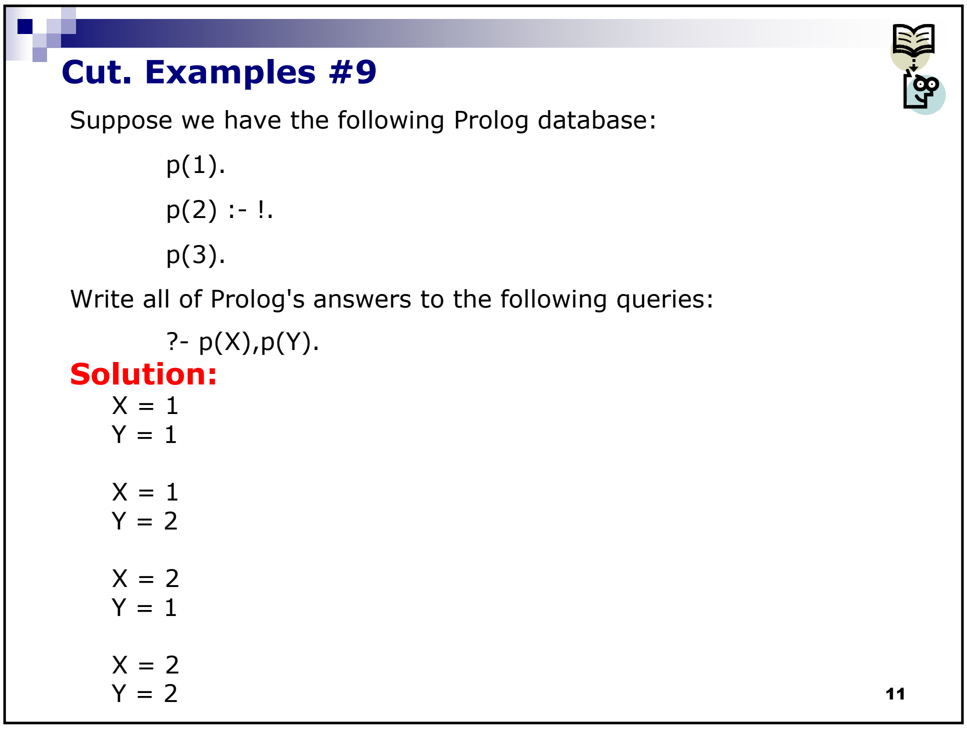
```
p(1).
p(2) :- !.
p(3).
```

Write all of Prolog's answers to the following queries:

```
?- p(X).
```

Solution:

```
X = 1;
X = 2
```



Cut. Examples #9

Suppose we have the following Prolog database:

```
p(1).
p(2) :- !.
p(3).
```

Write all of Prolog's answers to the following queries:

```
?- p(X),p(Y).
```

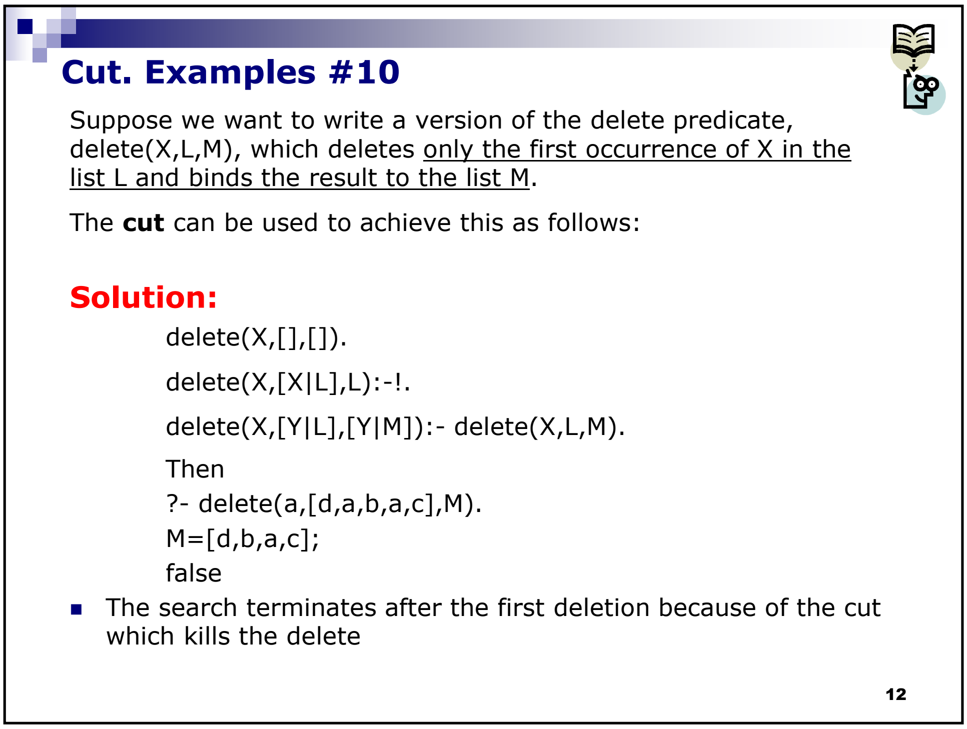
Solution:

```
X = 1
Y = 1

X = 1
Y = 2

X = 2
Y = 1

X = 2
Y = 2
```



Cut. Examples #10

Suppose we want to write a version of the delete predicate, delete(X,L,M), which deletes only the first occurrence of X in the list L and binds the result to the list M.

The **cut** can be used to achieve this as follows:

Solution:

```
delete(X,[],[]).
delete(X,[X|L],L):-!.
delete(X,[Y|L],[Y|M]):- delete(X,L,M).
Then
?- delete(a,[d,a,b,a,c],M).
M=[d,b,a,c];
false
```

- The search terminates after the first deletion because of the cut which kills the delete

Examples #11

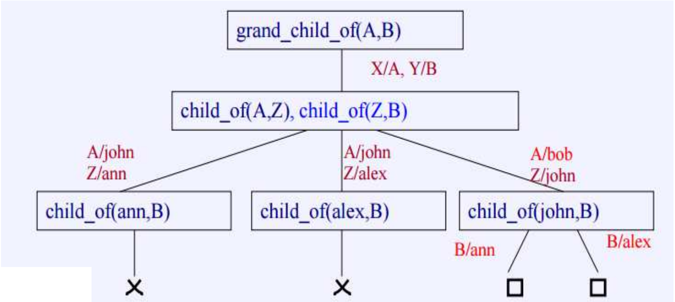
How PROLOG derives answers to the query

```
?- grand_child_of(A,B).
```

using the following program. Will any backtracking occur? Justify your answer using the corresponding derivation diagram.

```
child_of(john,ann).
child_of(john, alex).
child_of(bob, john).
grand_child_of(X,Y) :- child_of(X,Z), child_of(Z,Y).
```

Solution:



Examples #12

The following program says that two people are relatives if:

(a) one is a predecessor of the other, or
(b) they have a common predecessor, o r
(c) they have a common successor:

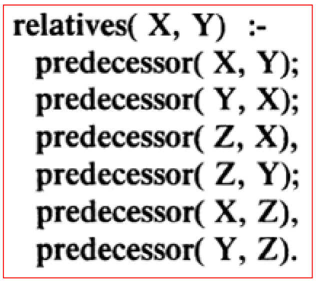
```
relatives( X,Y) :-
          predecessor( X,Y).

relatives( X,Y) :-
          predecessor(Y, X).

relatives(X,Y) :-
          predecessor(Z,X),
          predecessor(Z,Y).

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

Solution:



```
relatives( X, Y)  :-
   predecessor( X, Y);
   predecessor( Y, X);
   predecessor( Z, X),
   predecessor( Z, Y);
   predecessor( X, Z),
   predecessor( Y, Z).
```

Can you shorten this program by using the combination of coma and semicolon notation?