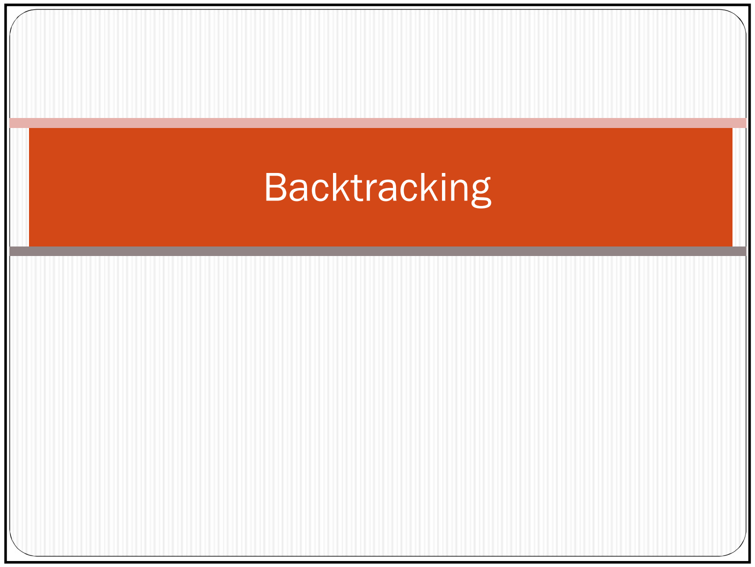
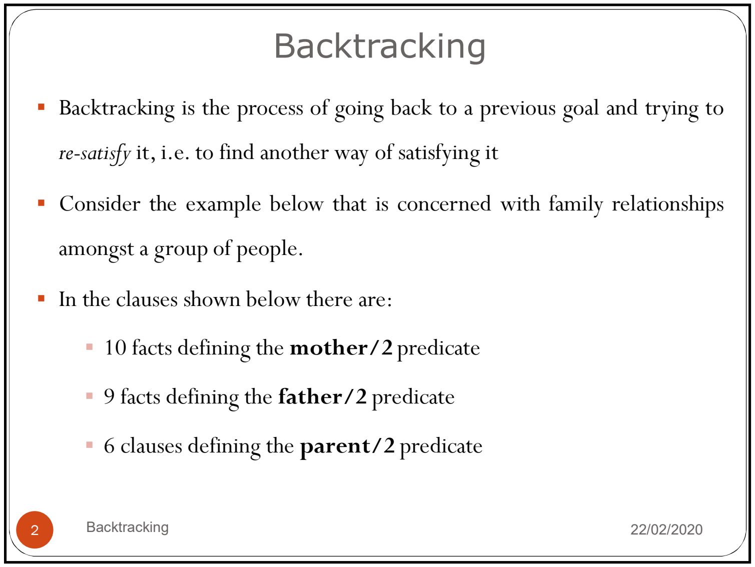
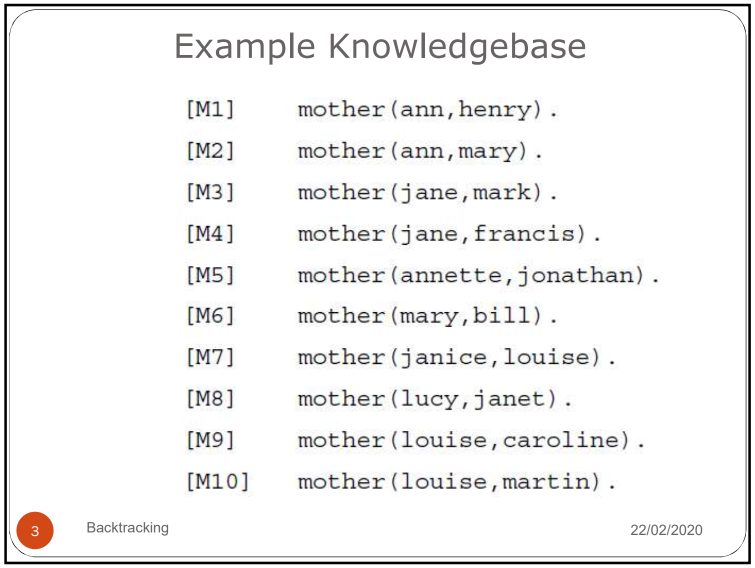


Backtracking



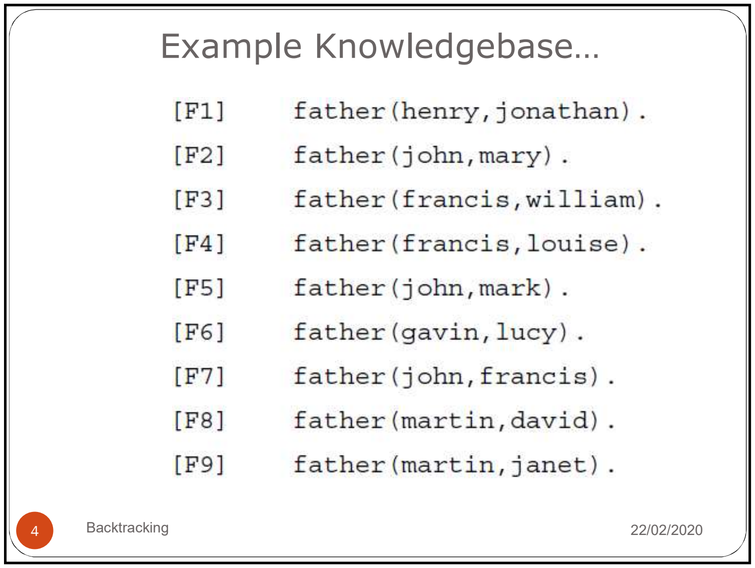
Backtracking

- Backtracking is the process of going back to a previous goal and trying to *re-satisfy* it, i.e. to find another way of satisfying it
- Consider the example below that is concerned with family relationships amongst a group of people.
- In the clauses shown below there are:
  - 10 facts defining the **mother/2** predicate
  - 9 facts defining the **father/2** predicate
  - 6 clauses defining the **parent/2** predicate



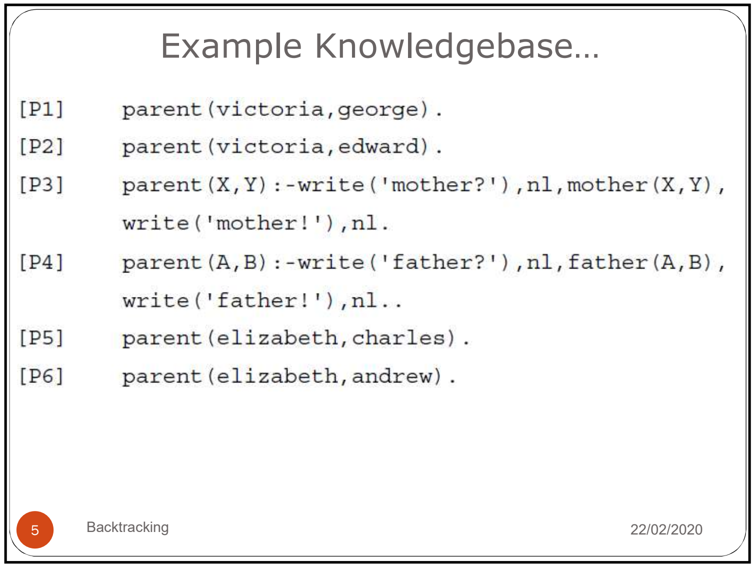
Example Knowledgebase

```
[M1]     mother(ann,henry).
[M2]     mother(ann,mary).
[M3]     mother(jane,mark).
[M4]     mother(jane,francis).
[M5]     mother(annette,jonathan).
[M6]     mother(mary,bill).
[M7]     mother(janice,louise).
[M8]     mother(lucy,janet).
[M9]     mother(louise,caroline).
[M10]    mother(louise,martin).
```



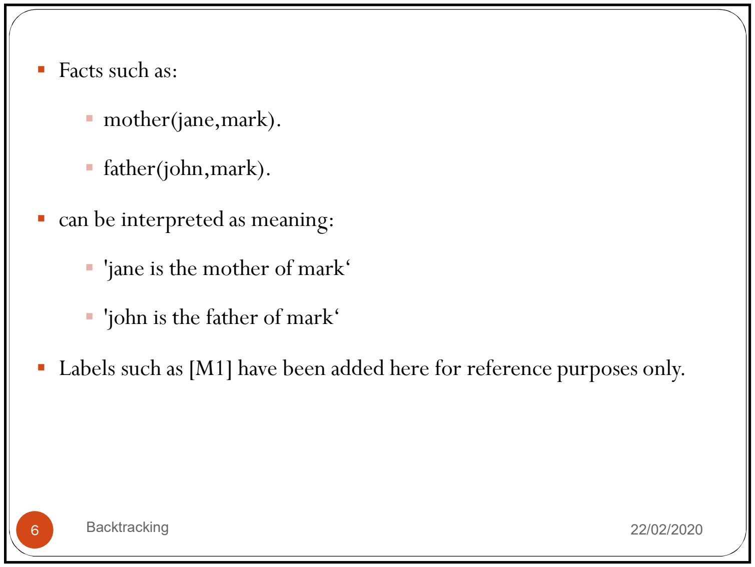
Example Knowledgebase…

```
[F1]     father(henry,jonathan).
[F2]     father(john,mary).
[F3]     father(francis,william).
[F4]     father(francis,louise).
[F5]     father(john,mark).
[F6]     father(gavin,lucy).
[F7]     father(john,francis).
[F8]     father(martin,david).
[F9]     father(martin,janet).
```

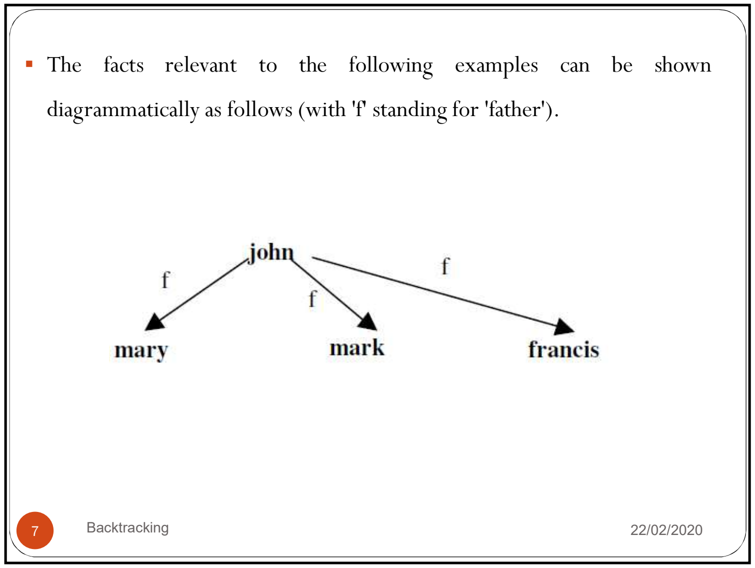


Example Knowledgebase…

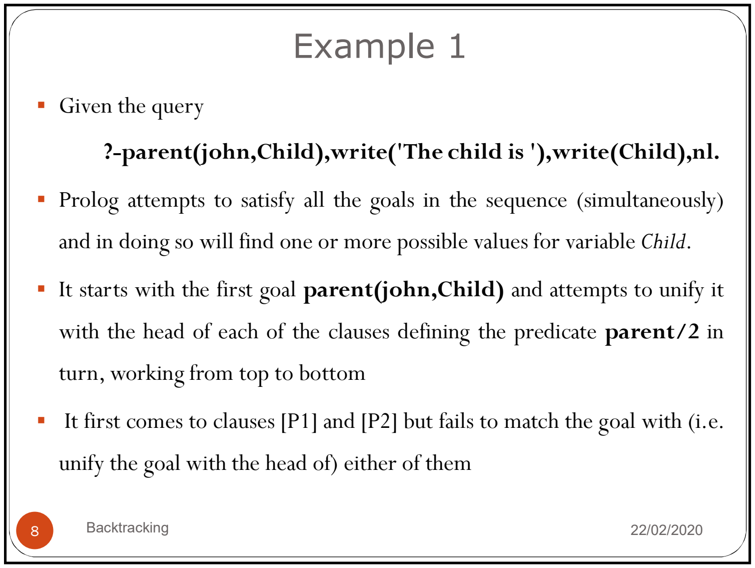
```
[P1]    parent(victoria,george).
[P2]    parent(victoria,edward).
[P3]    parent(X,Y):-write('mother?'),nl,mother(X,Y),
        write('mother!'),nl.
[P4]    parent(A,B):-write('father?'),nl,father(A,B),
        write('father!'),nl..
[P5]    parent(elizabeth,charles).
[P6]    parent(elizabeth,andrew).
```



- Facts such as:
  - mother(jane,mark).
  - father(john,mark).
- can be interpreted as meaning:
  - 'jane is the mother of mark‘
  - 'john is the father of mark‘
- Labels such as [M1] have been added here for reference purposes only.



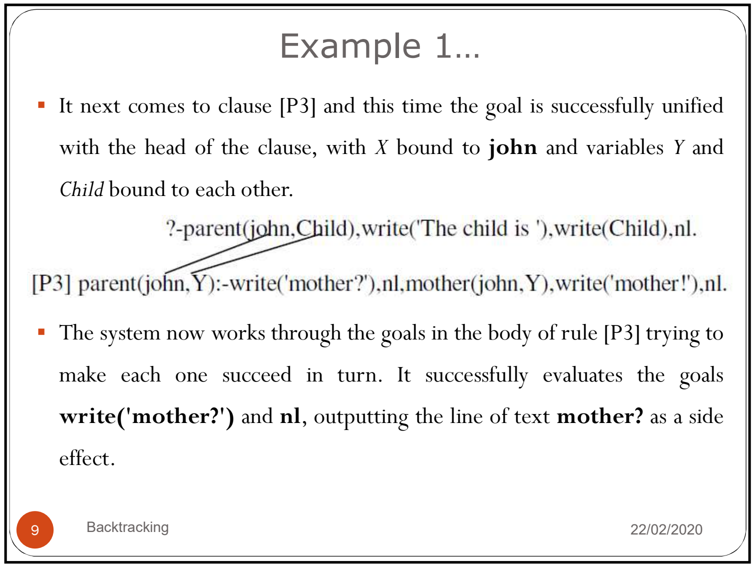
- The facts relevant to the following examples can be shown diagrammatically as follows (with 'f' standing for 'father').



Example 1

- Given the query

  ?-parent(john,Child),write('The child is '),write(Child),nl.
- Prolog attempts to satisfy all the goals in the sequence (simultaneously) and in doing so will find one or more possible values for variable *Child*.
- It starts with the first goal **parent(john,Child)** and attempts to unify it with the head of each of the clauses defining the predicate **parent/2** in turn, working from top to bottom
- It first comes to clauses [P1] and [P2] but fails to match the goal with (i.e. unify the goal with the head of) either of them

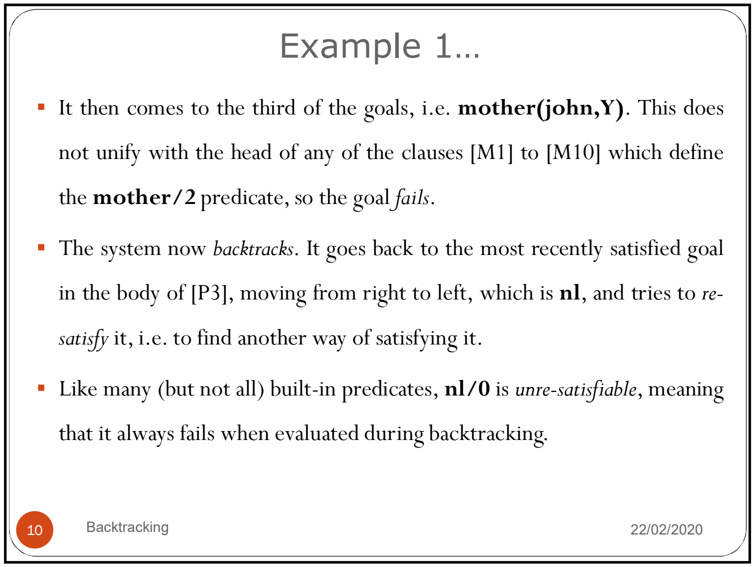


Example 1…

- It next comes to clause [P3] and this time the goal is successfully unified with the head of the clause, with *X* bound to **john** and variables *Y* and *Child* bound to each other.

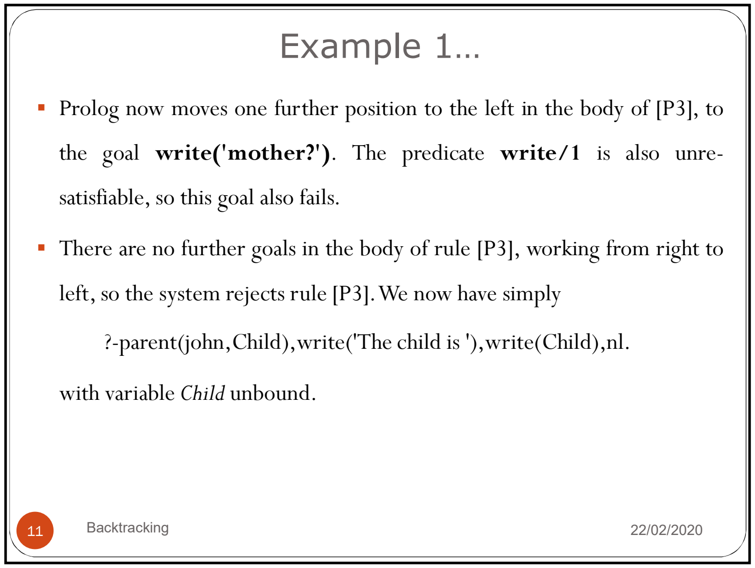
```
            ?-parent(john,Child),write('The child is '),write(Child),nl.

[P3] parent(john,Y):-write('mother?'),nl,mother(john,Y),write('mother!'),nl.
```

- The system now works through the goals in the body of rule [P3] trying to make each one succeed in turn. It successfully evaluates the goals **write('mother?')** and **nl**, outputting the line of text **mother?** as a side effect.



Example 1…

- It then comes to the third of the goals, i.e. **mother(john,Y)**. This does not unify with the head of any of the clauses [M1] to [M10] which define the **mother/2** predicate, so the goal *fails*.
- The system now *backtracks*. It goes back to the most recently satisfied goal in the body of [P3], moving from right to left, which is **nl**, and tries to *re-satisfy* it, i.e. to find another way of satisfying it.
- Like many (but not all) built-in predicates, **nl/0** is *unre-satisfiable*, meaning that it always fails when evaluated during backtracking.

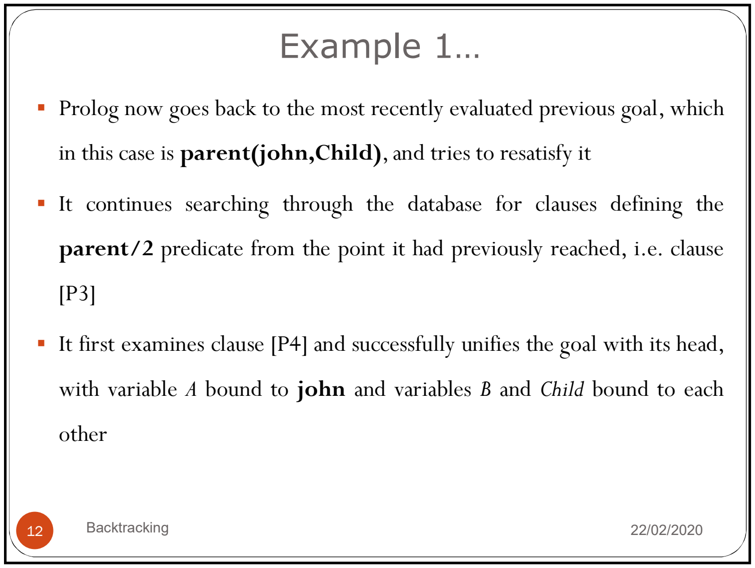


Example 1…

- Prolog now moves one further position to the left in the body of [P3], to the goal **write('mother?')**. The predicate **write/1** is also unre-satisfiable, so this goal also fails.
- There are no further goals in the body of rule [P3], working from right to left, so the system rejects rule [P3]. We now have simply

```
    ?-parent(john,Child),write('The child is '),write(Child),nl.
```

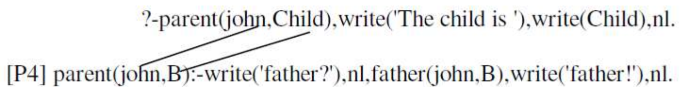
with variable *Child* unbound.



Example 1…

- Prolog now goes back to the most recently evaluated previous goal, which in this case is **parent(john,Child)**, and tries to resatisfy it
- It continues searching through the database for clauses defining the **parent/2** predicate from the point it had previously reached, i.e. clause [P3]
- It first examines clause [P4] and successfully unifies the goal with its head, with variable *A* bound to **john** and variables *B* and *Child* bound to each other

Example 1…



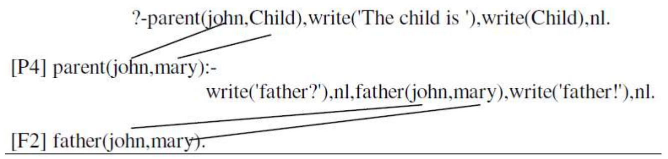
```
?-parent(john,Child),write('The child is '),write(Child),nl.
[P4] parent(john,B):-write('father?'),nl,father(john,B),write('father!'),nl.
```

- The system now works through the goals in the body of the rule [P4] trying to make each succeed in turn. The first two goals succeed, with the line of text, **father?** output as a side effect.

Example 1…

- The system now tries to satisfy the third goal, i.e. **father(john,B)**. It searches through the clauses defining the **father/2** predicate in turn, from top to bottom.
- The first clause it matches is [F2], which is a fact. This causes variable *B* to be bound to atom **mary**.
- This in turn causes variable *Child* (which is bound to variable *B*) to be bound to atom **mary**.

Example 1…



```
?-parent(john,Child),write('The child is '),write(Child),nl.
[P4] parent(john,mary):-
                write('father?'),nl,father(john,mary),write('father!'),nl.
[F2] father(john,mary).
```

Example 1…

- There are two further goals in the body of rule [P4], i.e. **write('father!')** and **nl**.
- These both succeed with the line of text **father!** output as a side effect.
- All the goals in the body of [P4] have now succeeded, so the head of the clause, which in rewritten form is **parent(john,mary)**, succeeds.
- The goal **parent(john,Child)** in the user's query therefore succeeds.

Example 1...

- The first of the goals in the sequence entered by the user has now been satisfied.
- There are three more goals in the sequence: **write('The child is ')**, **write(Child)** and **nl**.
- They all succeed, as a side effect outputting the line of text: **The child is mary**
- All the goals in the user's query have now been satisfied
- The Prolog system outputs the value of all the variables used in the query. In this case, the only one is *Child*.

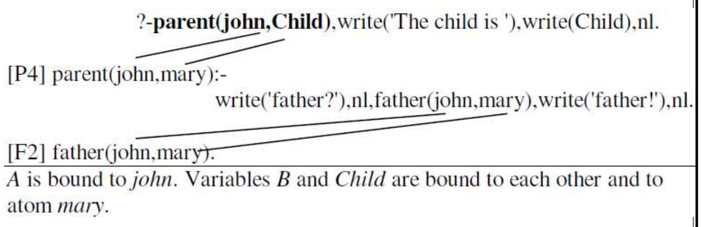
Example 1...

```
?- parent(john,Child),write('The child is '),write(Child),nl.
mother?
father?
father!
The child is mary
Child = mary
```

Forcing the System to Backtrack to Find Further Solutions

- The user can now force the system to backtrack to find a further solution or solutions by entering a semicolon character
- This works by forcing the most recently satisfied goal, i.e. **nl** (the last goal in the user's query) to fail. The system now backtracks to the previous goal in the sequence, i.e. **write(Child)**
- This too fails on backtracking, as does the previous goal, i.e. **write('The child is ')**
- The system backtracks a further step, to the first goal in the query, which is **parent(john,Child)**.

Forcing the System to Backtrack to Find Further Solutions...



```
        ?-parent(john,Child),write('The child is '),write(Child),nl.
[P4] parent(john,mary):-
          write('father?'),nl,father(john,mary),write('father!'),nl.
[F2] father(john,mary).
```

A is bound to *john*. Variables *B* and *Child* are bound to each other and to atom *mary*.

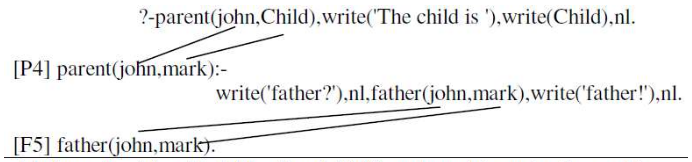
Forcing the System to Backtrack to Find Further Solutions…

- The system attempts to find another way of satisfying it, beginning by trying to find another way of satisfying the last goal in the body of [P4]
- This is **nl**, which fails on backtracking
- So too does the previous goal **write('father!')**.
- It now attempts to resatisfy the previous goal in the body of [P4], working from right to left, which is **father(john,B)**
- This process begins by rejecting the unification with the head of [F2]
- Prolog now continues to search through the clauses defining the **father/2** predicate for further unifications

Forcing the System to Backtrack to Find Further Solutions…

- The next successful unification is with the head of clause [F5]. The terms **father(john,B)** and **father(john,mark)** are unified with variable *B* bound to **mark**
- This causes variable *Child* also to be bound to **mark**.

Forcing the System to Backtrack to Find Further Solutions…



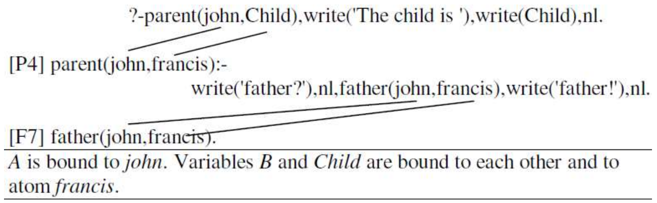
```
        ?-parent(john,Child),write('The child is '),write(Child),nl.

[P4] parent(john,mark):-
              write('father?'),nl,father(john,mark),write('father!'),nl.

[F5] father(john,mark).
```

A is bound to *john*. Variables *B* and *Child* are bound to each other and to atom *mark*.

Forcing the System to Backtrack to Find Further Solutions…

- This gives a second solution to the user's goal, i.e. a second way of satisfying it.
- Further backtracking will find a third solution, using clause [F7].

Forcing the System to Backtrack to Find Further Solutions…



```
?-parent(john,Child),write('The child is '),write(Child),nl.

[P4] parent(john,francis):-
          write('father?'),nl,father(john,francis),write('father!'),nl.

[F7] father(john,francis).
```

A is bound to *john*. Variables *B* and *Child* are bound to each other and to atom *francis*.

```
?- parent(john,Child),write('The child is '),write(Child),nl.
mother?
father?
```

Forcing the System to Backtrack to Find Further Solutions…

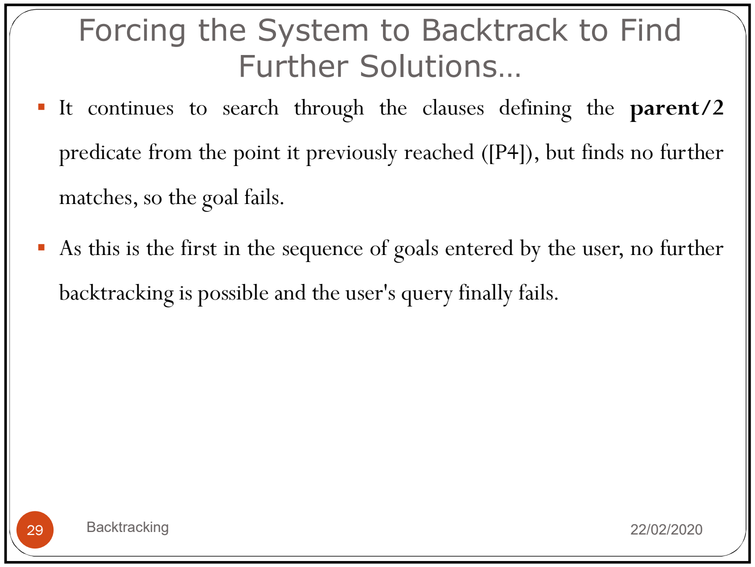
```
?- parent(john,Child),write('The child is '),write(Child),nl.
mother?
father?
father!
The child is mary
Child = mary ;
father!
The child is mark
Child = mark ;
father!
The child is francis
Child = francis
```

Forcing the System to Backtrack to Find Further Solutions…

- If the user again enters a semicolon to force the system to backtrack, the system will again go through the backtracking sequence described above, until it reaches the stage of attempting to re-satisfy **father(john,B)**, by rejecting the unification with the head of clause [F7] previously found and continuing to search through the clauses defining the **father/2** predicate for further matches
- As no further unifications are found, the goal **father(john,B)** in the body of rule [P4] will now fail.

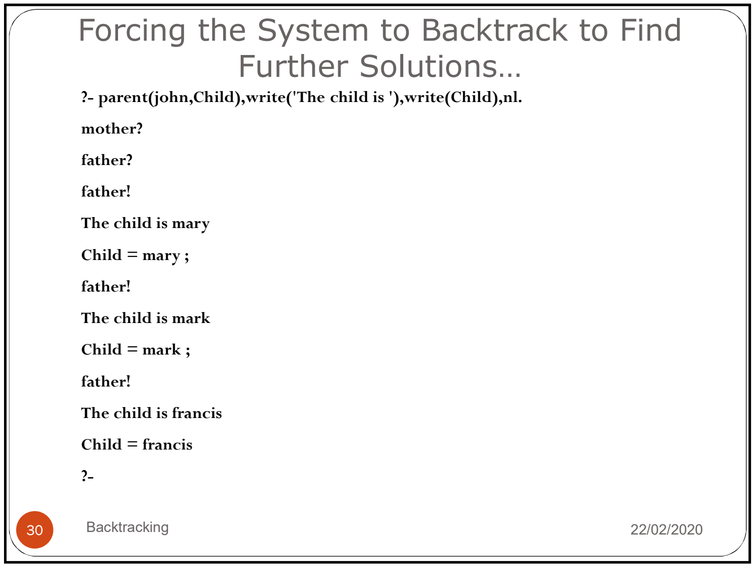
Forcing the System to Backtrack to Find Further Solutions…

- The system now attempts to re-satisfy the goal to the left of it in the body of rule [P4].
- This is **nl**, which always fails on backtracking
- The next goal, again moving to the left, is **write('father?')**, which also fails. There are no further goals in the body of [P4], moving from right to left, so the system rejects rule [P4]
- This brings it back to the original goal **parent(john,Child)**, which it tries to re-satisfy.



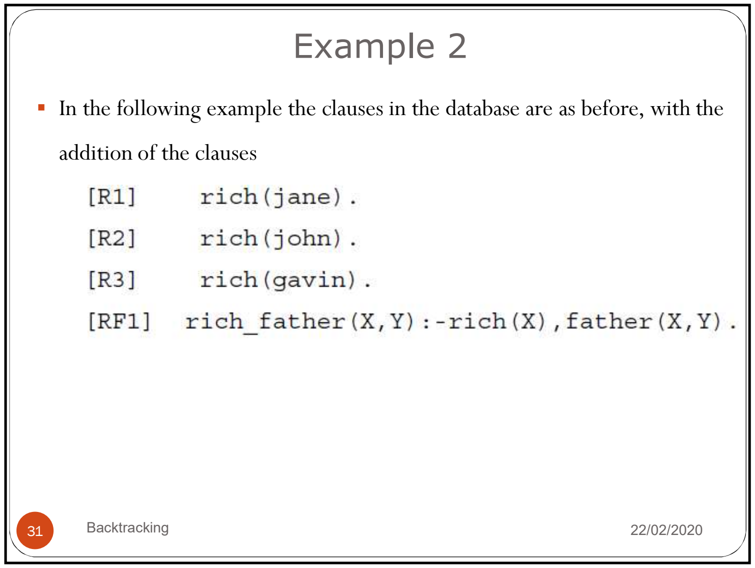
Forcing the System to Backtrack to Find Further Solutions…

- It continues to search through the clauses defining the **parent/2** predicate from the point it previously reached ([P4]), but finds no further matches, so the goal fails.
- As this is the first in the sequence of goals entered by the user, no further backtracking is possible and the user's query finally fails.



Forcing the System to Backtrack to Find Further Solutions…

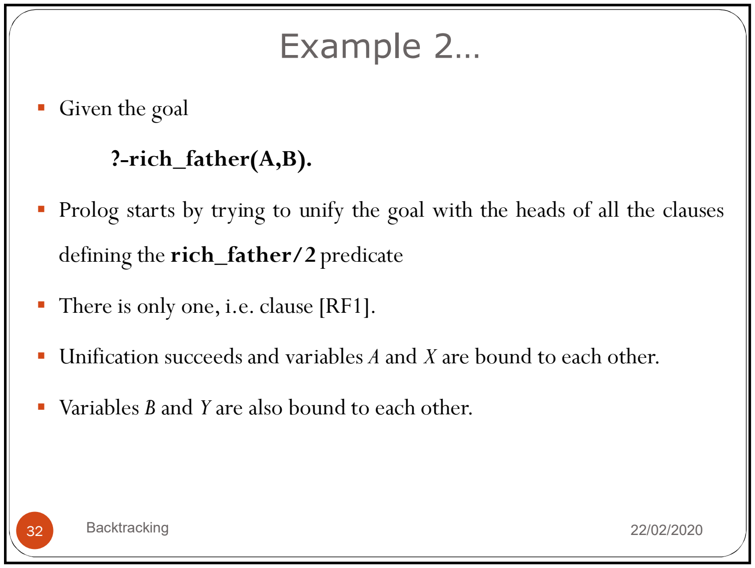
```
?- parent(john,Child),write('The child is '),write(Child),nl.
mother?
father?
father!
The child is mary
Child = mary ;
father!
The child is mark
Child = mark ;
father!
The child is francis
Child = francis
?-
```



Example 2

- In the following example the clauses in the database are as before, with the addition of the clauses

```
[R1]     rich(jane).
[R2]     rich(john).
[R3]     rich(gavin).
[RF1]   rich_father(X,Y):-rich(X),father(X,Y).
```



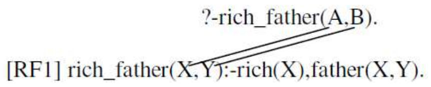
Example 2…

- Given the goal

  ?-rich_father(A,B).

- Prolog starts by trying to unify the goal with the heads of all the clauses defining the **rich_father/2** predicate
- There is only one, i.e. clause [RF1].
- Unification succeeds and variables *A* and *X* are bound to each other.
- Variables *B* and *Y* are also bound to each other.

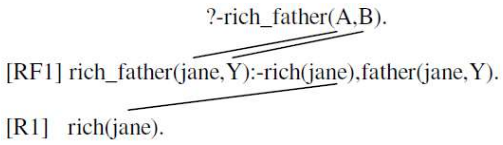
Example 2...


?-rich_father(A,B).

[RF1] rich_father(X,Y):-rich(X),father(X,Y).

- Next Prolog tries to find a value of *A* satisfying the first goal in the body of rule [RF1]
- It does this by searching through the clauses defining the **rich/1** predicate.
- The first unification it finds is with the head of [R1], i.e. **rich(jane)**. *X* is bound to **jane**.

Example 2...


?-rich_father(A,B).

[RF1] rich_father(jane,Y):-rich(jane),father(jane,Y).

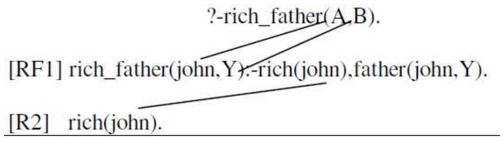
[R1] rich(jane).

- The system now tries to satisfy the goal **father(jane,Y)** by examining the clauses defining the **father/2** predicate, i.e. [F1] to [F9]
- None of them unify with the goal, so the system backtracks and attempts to re-satisfy solution to) the most recently satisfied goal, which is **rich(X)**

Example 2...

- It continues searching through the clauses defining the **rich/1** predicate, the next unification found being with **rich(john)** (clause [R2]). Now *X* is bound to **john**, which in turn causes *A* to be bound to **john**. (i.e. find another solution to) the most recently satisfied goal, which is **rich(X)**
- It continues searching through the clauses defining the **rich/1** predicate, the next unification found being with **rich(john)** (clause [R2]). Now *X* is bound to **john**, which in turn causes *A* to be bound to **john**.

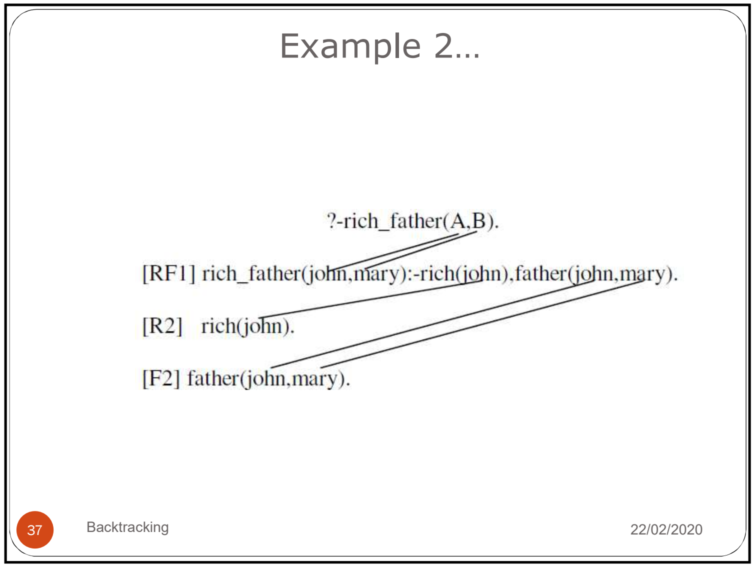
Example 2...


?-rich_father(A,B).

[RF1] rich_father(john,Y):-rich(john),father(john,Y).

[R2] rich(john).

- The system now tries to satisfy the goal **father(john,Y)** by examining the clauses defining the **father/2** predicate, i.e. [F1] to [F9]. The first unification found is with [F2], i.e. **father(john,mary)**. *Y* is bound to **mary**.



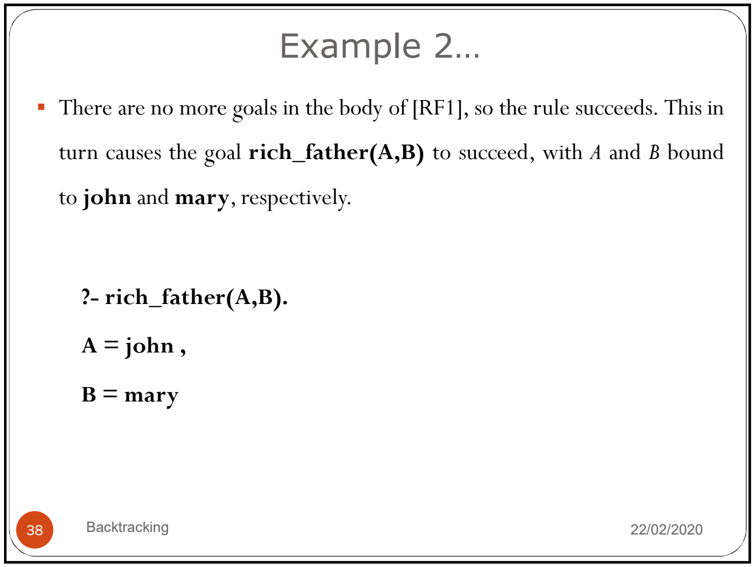
Example 2…

?-rich_father(A,B).

[RF1] rich_father(john,mary):-rich(john),father(john,mary).

[R2] rich(john).

[F2] father(john,mary).



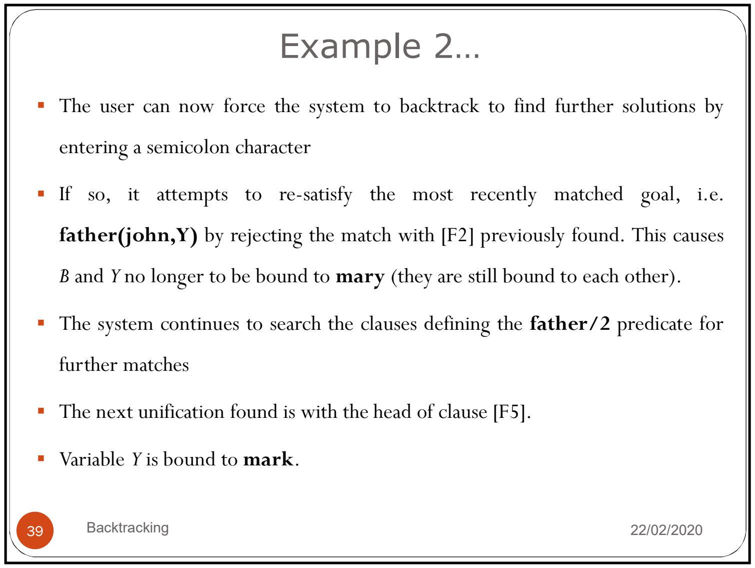
Example 2…

- There are no more goals in the body of [RF1], so the rule succeeds. This in turn causes the goal **rich_father(A,B)** to succeed, with *A* and *B* bound to **john** and **mary**, respectively.

?- rich_father(A,B).

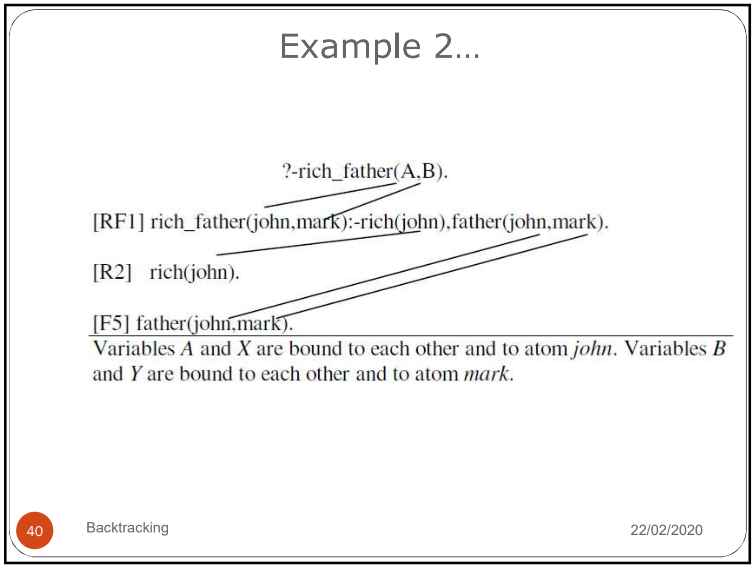
A = john ,

B = mary



Example 2…

- The user can now force the system to backtrack to find further solutions by entering a semicolon character
- If so, it attempts to re-satisfy the most recently matched goal, i.e. **father(john,Y)** by rejecting the match with [F2] previously found. This causes *B* and *Y* no longer to be bound to **mary** (they are still bound to each other).
- The system continues to search the clauses defining the **father/2** predicate for further matches
- The next unification found is with the head of clause [F5].
- Variable *Y* is bound to **mark**.



Example 2…

?-rich_father(A,B).

[RF1] rich_father(john,mark):-rich(john),father(john,mark).

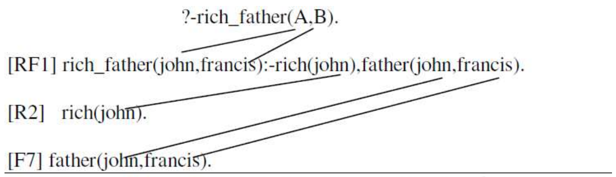
[R2] rich(john).

[F5] father(john,mark).

Variables *A* and *X* are bound to each other and to atom *john*. Variables *B* and *Y* are bound to each other and to atom *mark*.

Example 2…

- This gives a second solution to the user's query. If the user forces the system to backtrack again, it will find a third solution using clause [F7] **father(john,francis)**.



?-rich_father(A,B).

[RF1] rich_father(john,francis):-rich(john),father(john,francis).

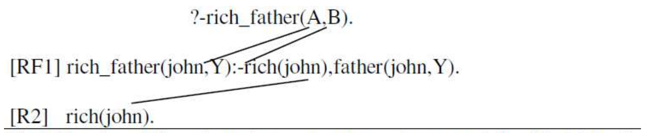
[R2] rich(john).

[F7] father(john,francis).

Variables *A* and *X* are bound to each other and to atom *john*. Variables *B* and *Y* are bound to each other and to atom *francis*.

Example 2…

- If the user forces the system to backtrack again, it will start by deeming that the most recently satisfied goal, i.e. **father(john,Y)** has failed
- This causes *B* and *Y* no longer to be bound to **francis** (they are still bound to each other).



?-rich_father(A,B).

[RF1] rich_father(john,Y):-rich(john),father(john,Y).

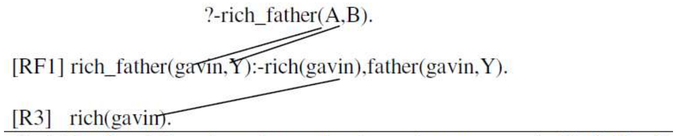
[R2] rich(john).

Variables *A* and *X* are bound to each other and to atom *john*. Variables *B* and *Y* are bound to each other.

Example 2…

- The system will fail to find any further matches for the goal **father(john,Y)**
- It will next attempt to find further solutions to the most recently satisfied previous goal in [RF1], working from right to left.
- This is **rich(X)**. This will succeed with *X* now bound to **gavin** (clause [R3]).

Example 2…



?-rich_father(A,B).

[RF1] rich_father(gavin,Y):-rich(gavin),father(gavin,Y).

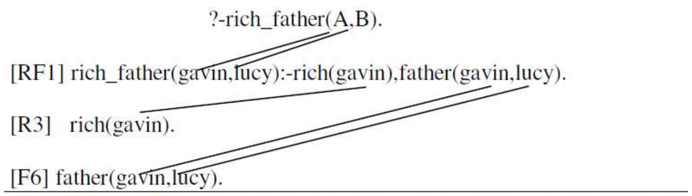
[R3] rich(gavin).

Variables *A* and *X* are bound to each other and to atom *gavin*. Variables *B* and *Y* are bound to each other.

Example 2...

- Working from left to right again, the system will now try to satisfy the goal **father(gavin,Y)**.
- This will unify with the head of just one of the **father/2** clauses, namely with clause [F6] **father(gavin,lucy)**, with variable *Y* bound to **lucy**.

Example 2...



?-rich_father(A,B).

[RF1] rich_father(gavin,lucy):-rich(gavin),father(gavin,lucy).

[R3] rich(gavin).

[F6] father(gavin,lucy).

Variables *A* and *X* are bound to each other and to atom *gavin*. Variables *B* and *Y* are bound to each other and to atom *lucy*.

Example 2...

- All the goals in the body of [RF1] have now succeeded, so the head **rich_father(gavin,lucy)** succeeds, and in turn **rich_father(A,B)** succeds with *A* and *B* bound to **gavin** and **lucy**, respectively.
- Forcing the system to backtrack again will lead to the same sequence of operations as above, right up to the attempt to find further matches for the goal **rich(X)** in the body of [PF1]
- This will fail, which will in turn cause [RF1] to fail.

Example 2...

- This will make the Prolog system go back a further step to try to find another match for the original goal **rich_father(A,B)** with clauses defining the**rich_father/2** predicate.
- Since there is only one such clause, no more matches will be found and the user's goal will finally fail.

Example 2…

?- rich_father(A,B).

A = john ,

B = mary ;

A = john ,

B = mark ;

A = john ,

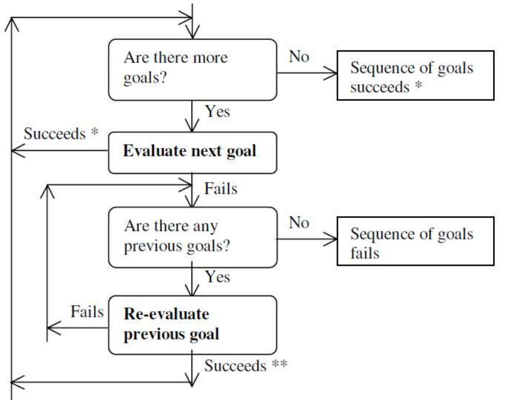
B = francis ;

A = gavin ,

B = lucy ;

?-

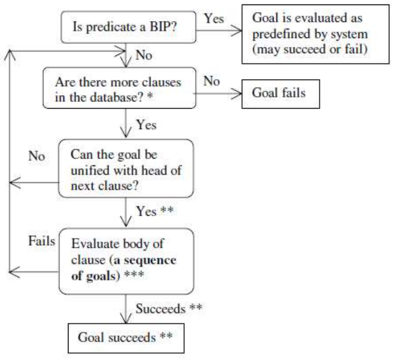
Satisfying Goals: A Summary



* Some variables may have become bound.

** Some variables may have become bound to other values (or unbound).

Evaluating a Sequence of Goals: Summary


Is predicate a BIP?
Yes
Goal is evaluated as predefined by system (may succeed or fail)
No
Are there more clauses in the database? *
No
Goal fails
Yes
No
Can the goal be unified with head of next clause?
Yes **
Fails
Evaluate body of clause (a sequence of goals) ***
Succeeds **
Goal succeeds **

* Evaluation: Start at top of database.
Re-evaluation: Start after clause matched when goal last satisfied.

** Some variables may have become bound.

*** If the clause is a fact there is no body, so the goal succeeds immediately.

END