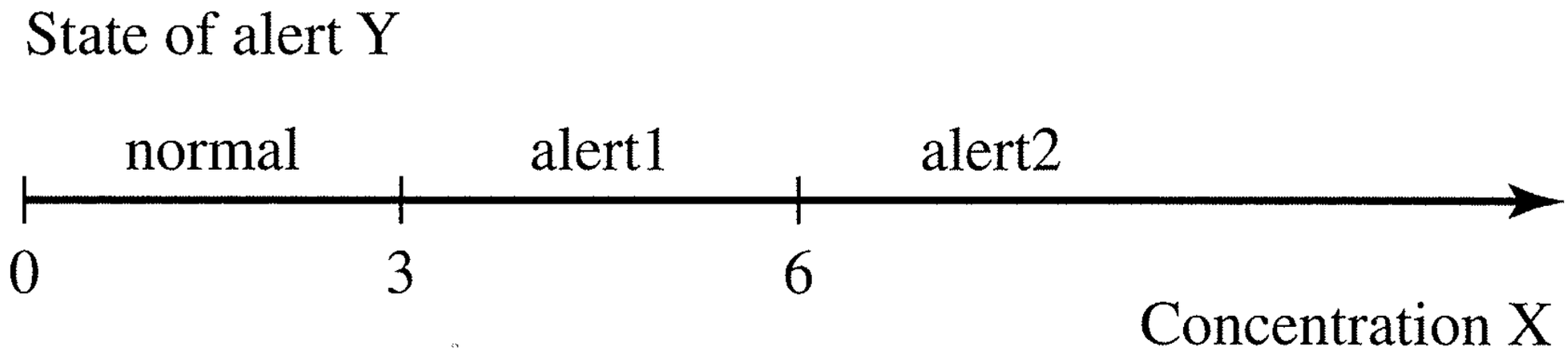


Figure 5.1 State of alert as a function of pollution level.

Rule 1: if $X < 3$ then Y = normal
Rule 2: if $3 \leq X$ and $X < 6$ then Y = alert1
Rule 3: if $6 \leq X$ then Y = alert2

```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```

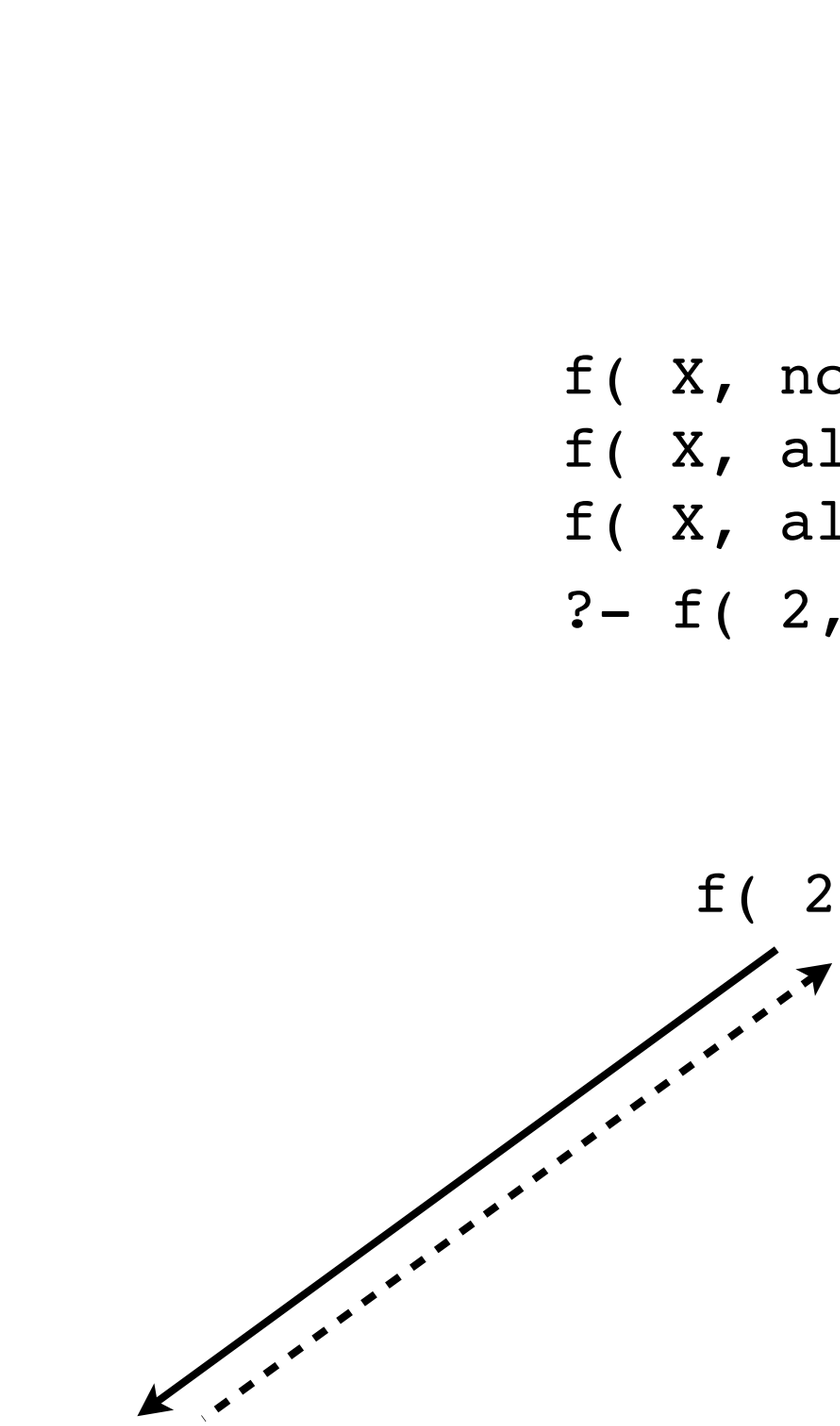
```
f( 2, Y), Y = alert1.
```

```
X < 3.
```

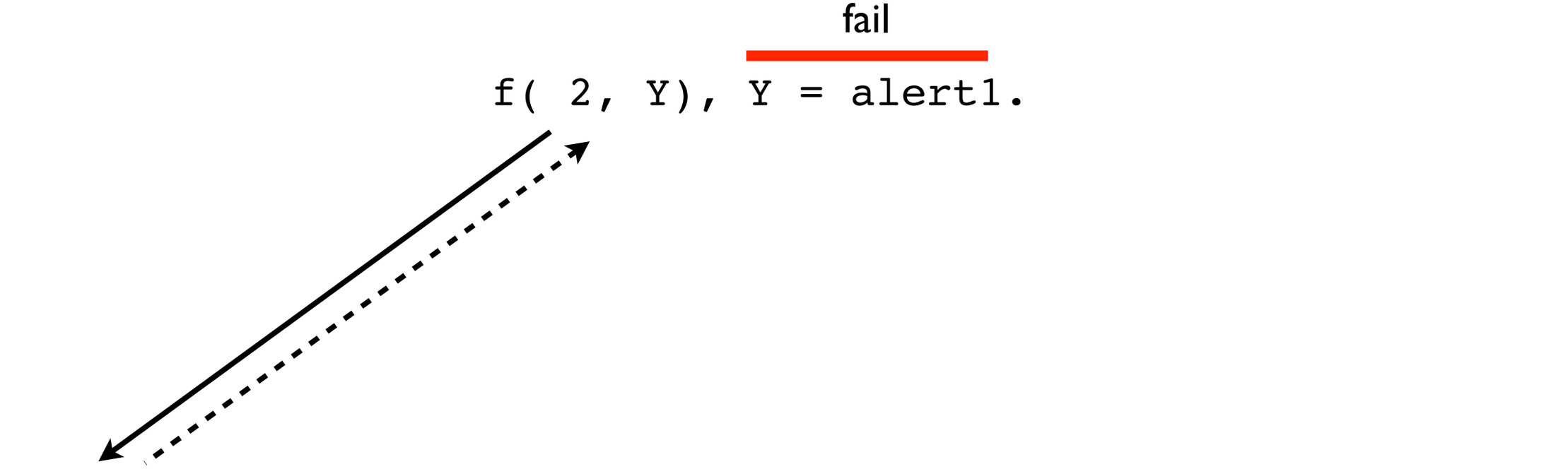
```
3 =< X, X < 6.
```

```
6 =< X.
```

```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```



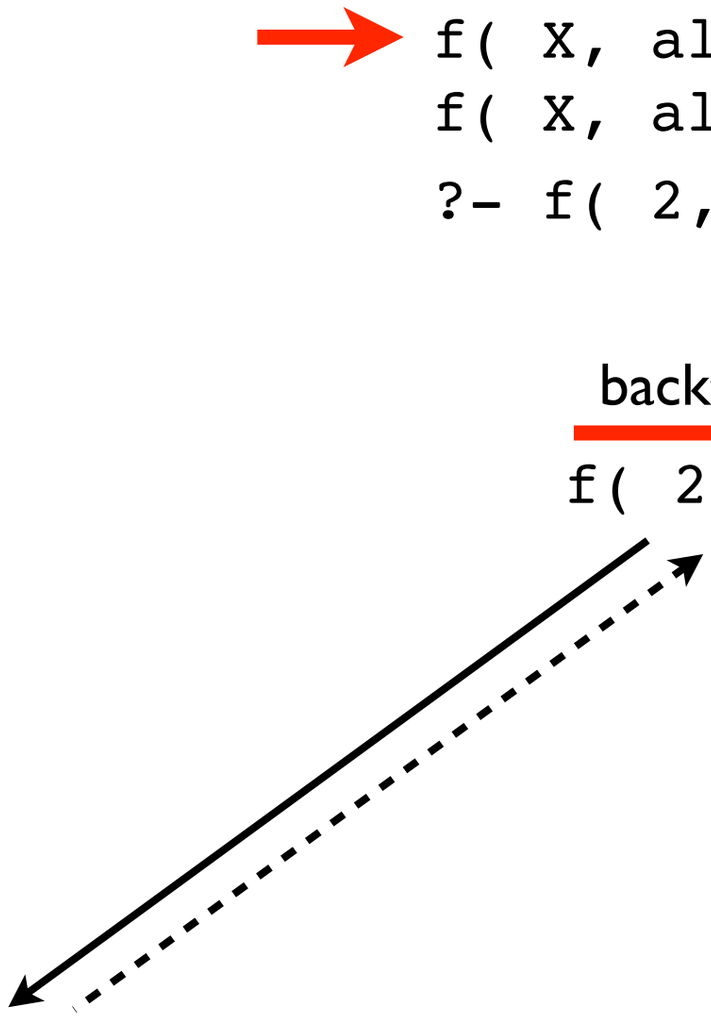
```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```



```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```

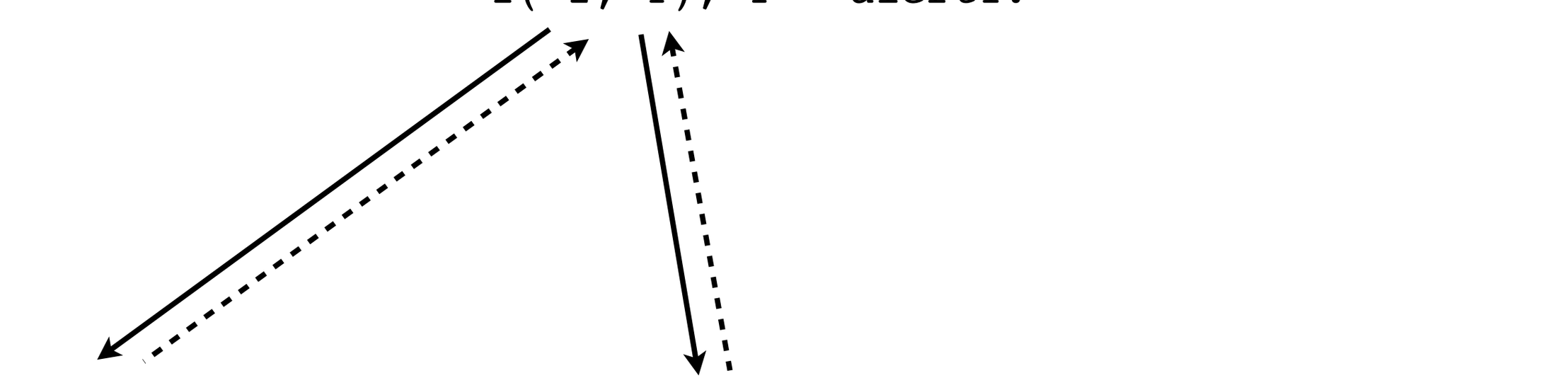
backtrack

```
f( 2, Y), Y = alert1.
```

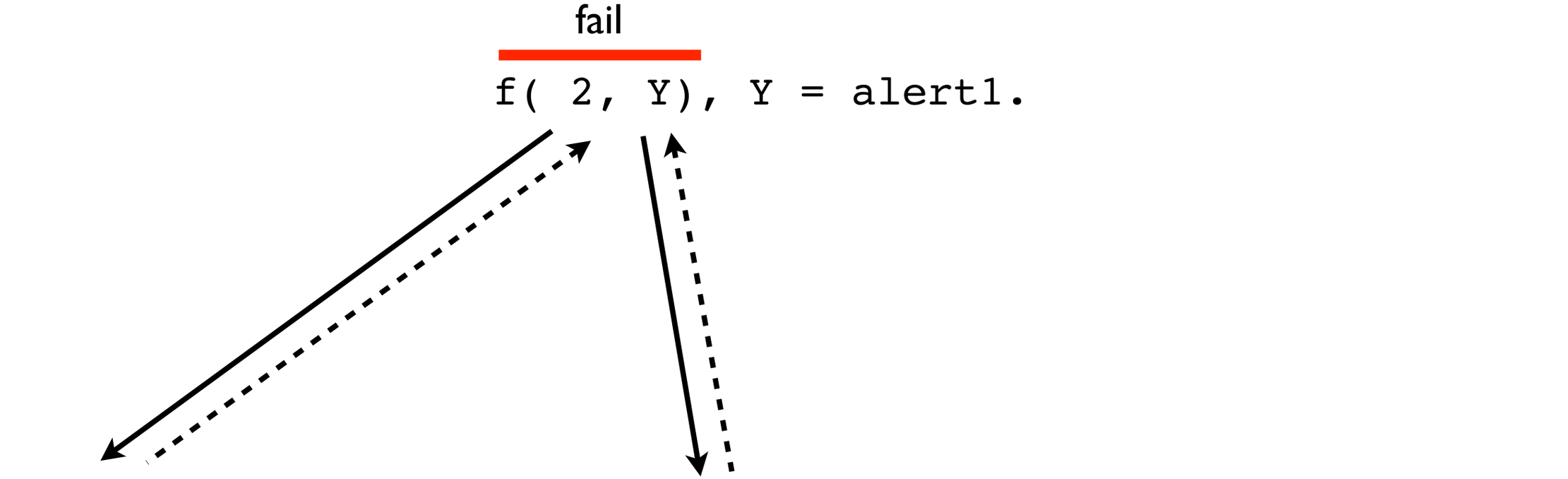


```
X < 3.          3 =< X, X < 6.          6 =< X.
```

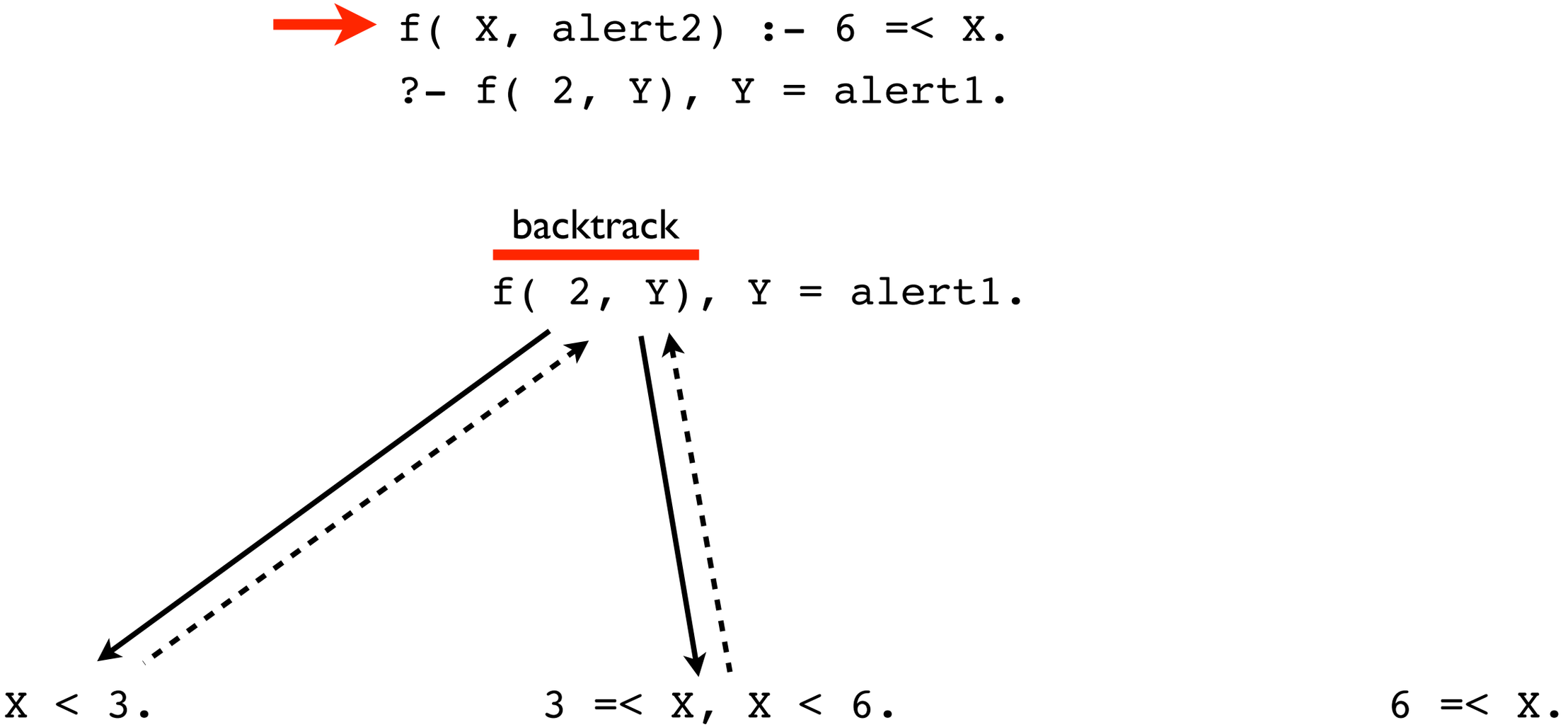
```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```



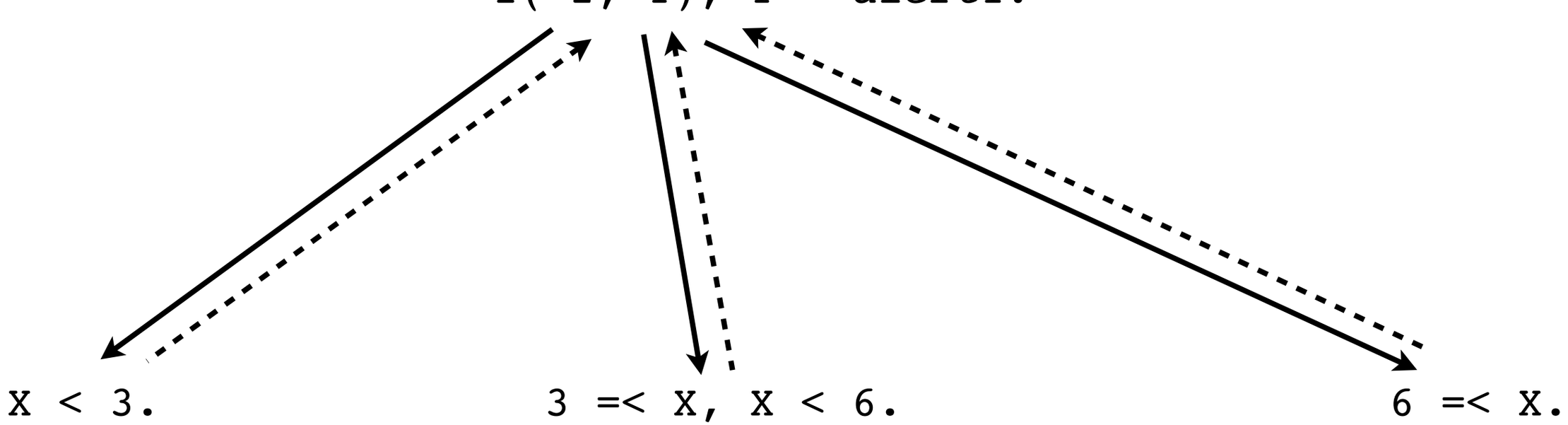
```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```



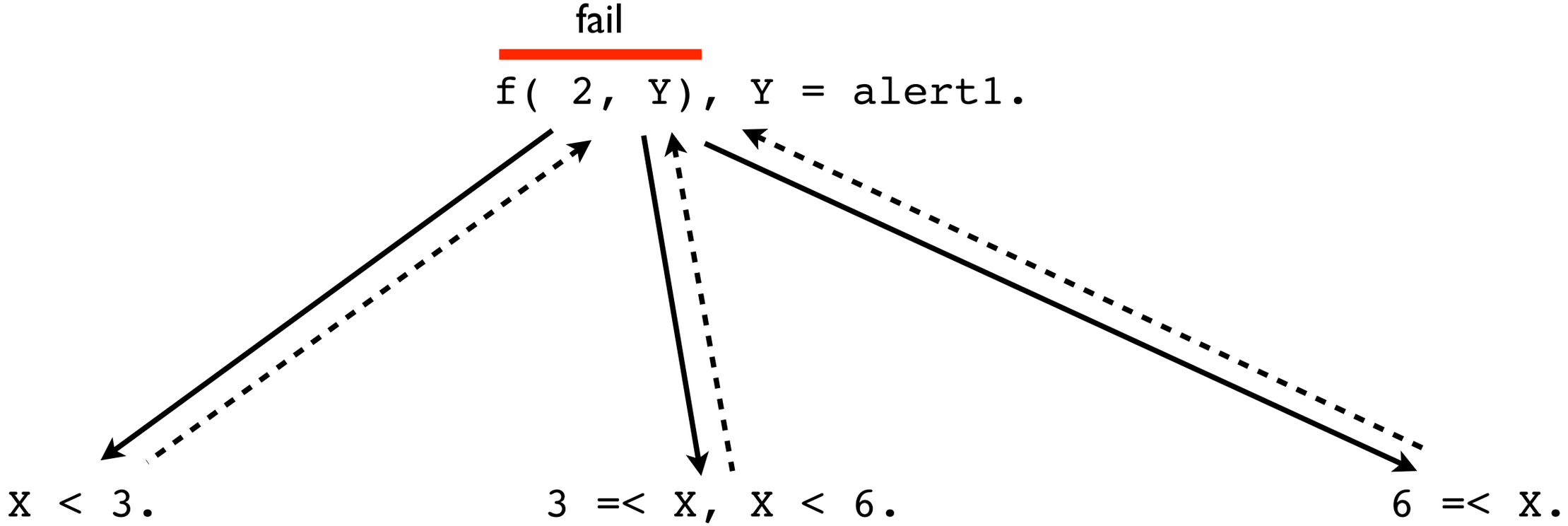
```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```



```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```



```
f( X, normal) :- X < 3.
f( X, alert1) :- 3 =< X, X < 6.
f( X, alert2) :- 6 =< X.
?- f( 2, Y), Y = alert1.
```



No cut in function `f`.
The cut in function `f2`.

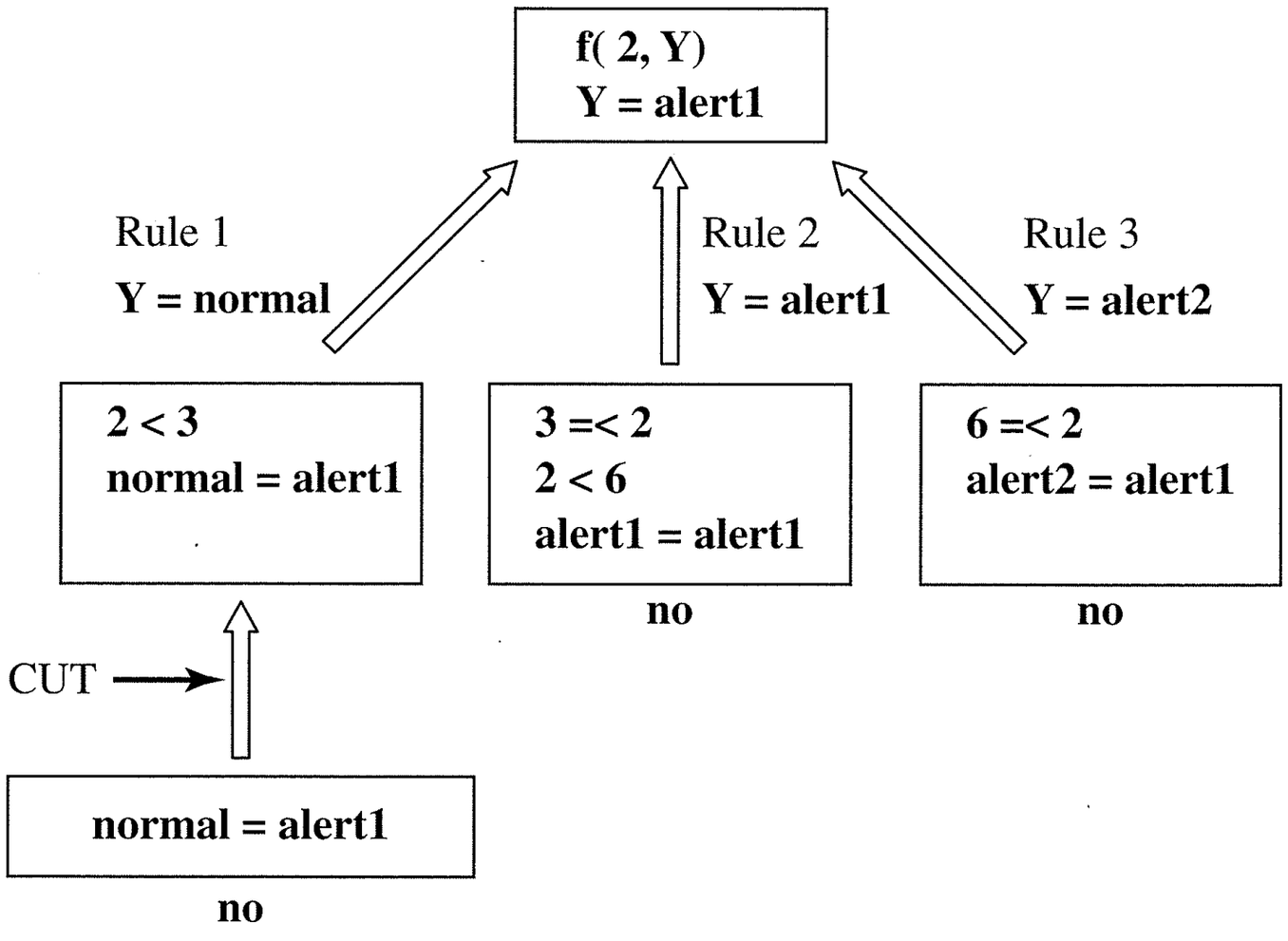


Figure 5.2 At the point marked 'CUT' we already know that the rules 2 and 3 are bound to fail.

```
f2( X, normal) :- X < 3, !.
f2( X, alert1) :- 3 =< X, X < 6, !.
f2( X, alert2) :- 6 =< X.
?- f2( 2, Y), Y = alert1.
```

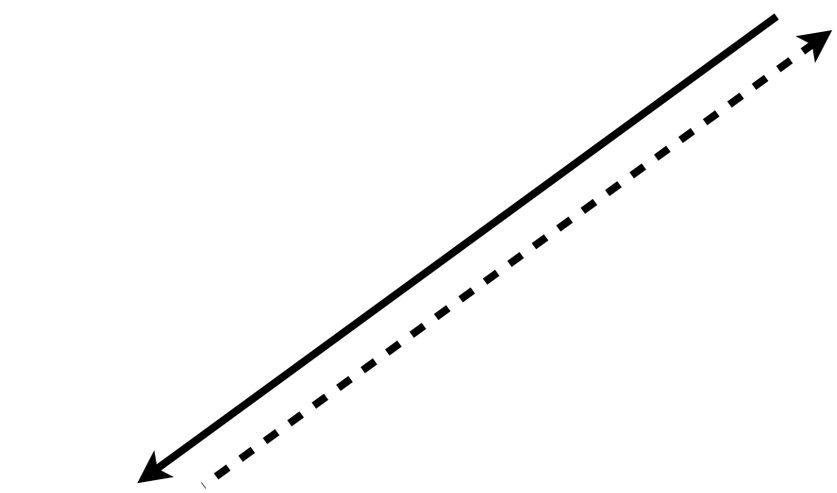
```
f2( 2, Y), Y = alert1.
```

```
X < 3, !.
```

```
3 =< X, X < 6, !.
```

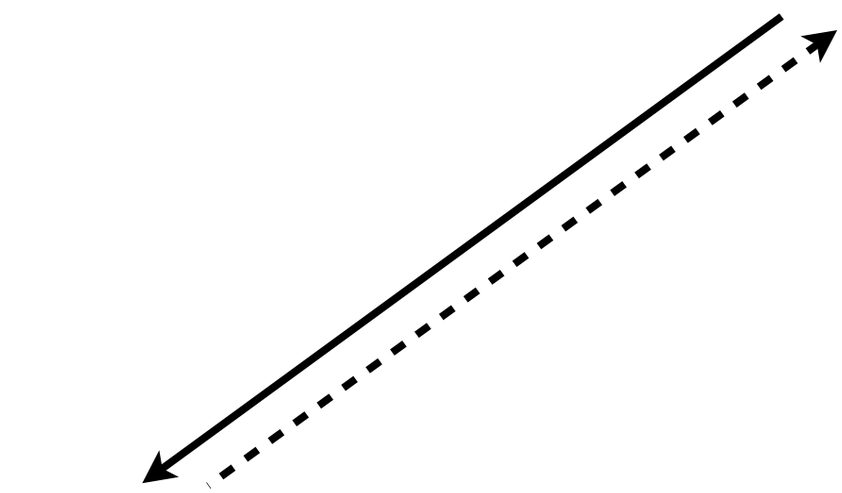
```
6 =< X.
```

```
f2( X, normal) :- X < 3, !.
f2( X, alert1) :- 3 =< X, X < 6, !.
f2( X, alert2) :- 6 =< X.
?- f2( 2, Y), Y = alert1.
```



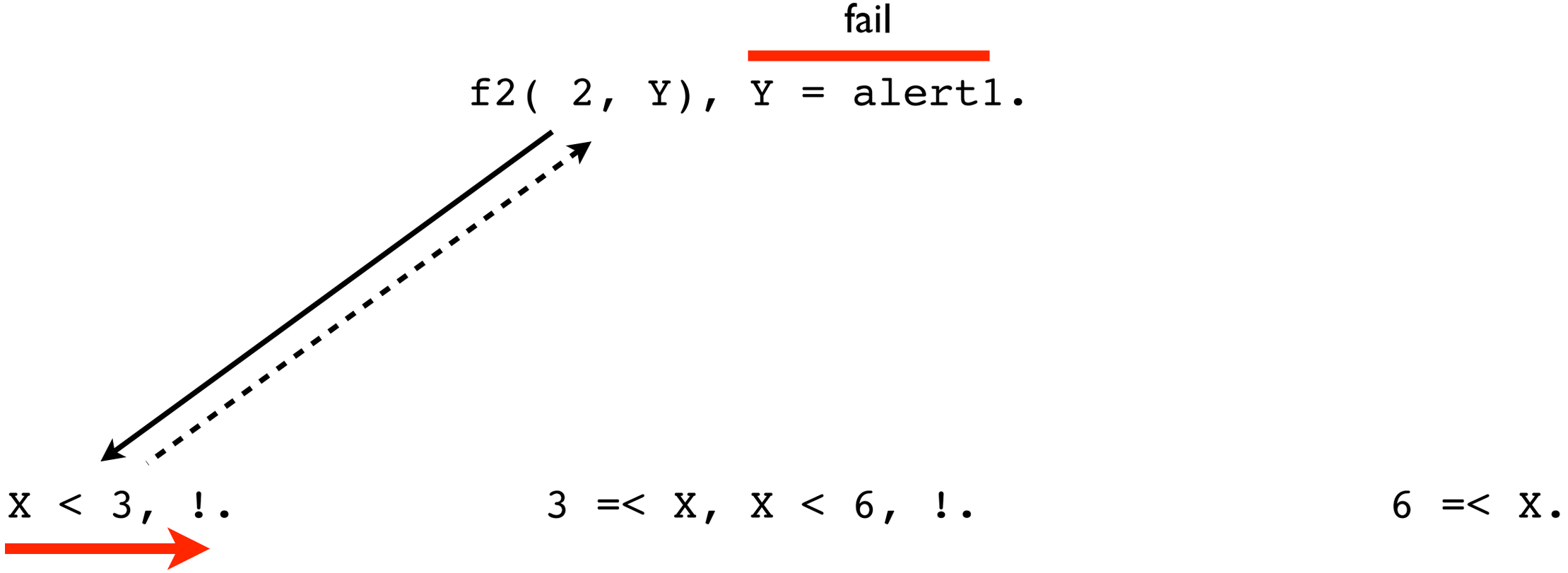

```
X < 3, !.          3 =< X, X < 6, !.          6 =< X.
```

```
f2( X, normal) :- X < 3, !.
f2( X, alert1) :- 3 =< X, X < 6, !.
f2( X, alert2) :- 6 =< X.
?- f2( 2, Y), Y = alert1.
```



Commits to this choice.

```
f2( X, normal) :- X < 3, !.
f2( X, alert1) :- 3 =< X, X < 6, !.
f2( X, alert2) :- 6 =< X.
?- f2( 2, Y), Y = alert1.
```



Commits to this choice.

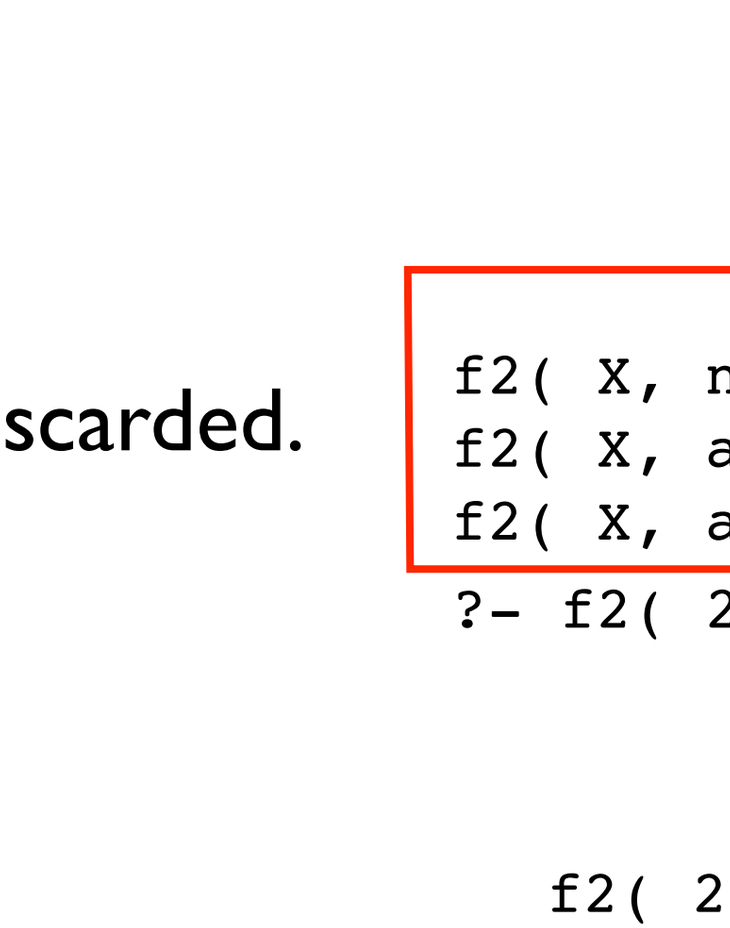
Discarded.

```
f2( X, normal) :- X < 3, !.
f2( X, alert1) :- 3 =< X, X < 6, !.
f2( X, alert2) :- 6 =< X.
```

```
?- f2( 2, Y), Y = alert1.
```

fail

```
f2( 2, Y), Y = alert1.
```

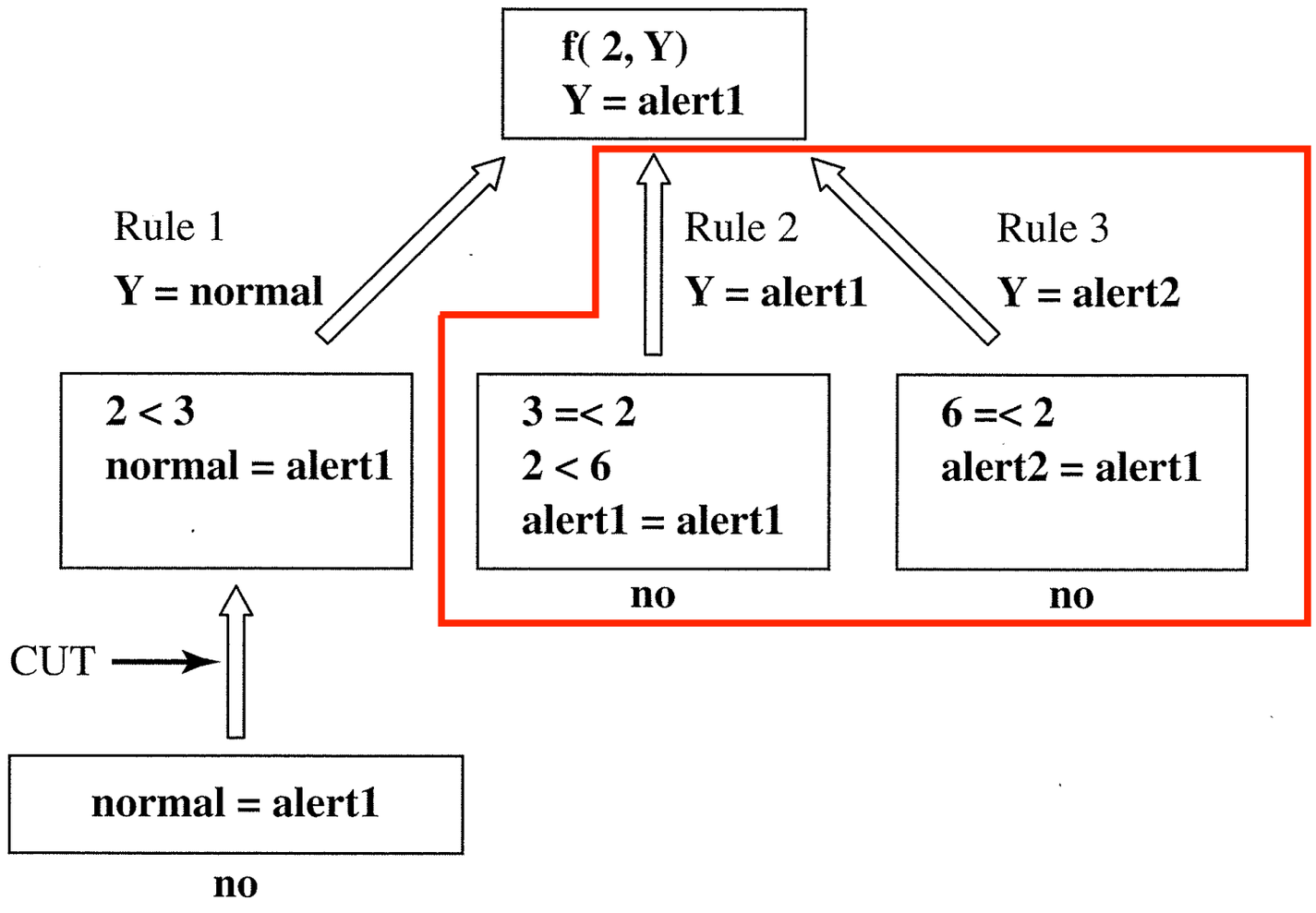


```
X < 3, !.
```

```
3 =< X, X < 6, !.
```

```
6 =< X.
```

No cut in function `f`.
The cut in function `f2`.



Discarded.

Figure 5.2 At the point marked 'CUT' we already know that the rules 2 and 3 are bound to fail.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

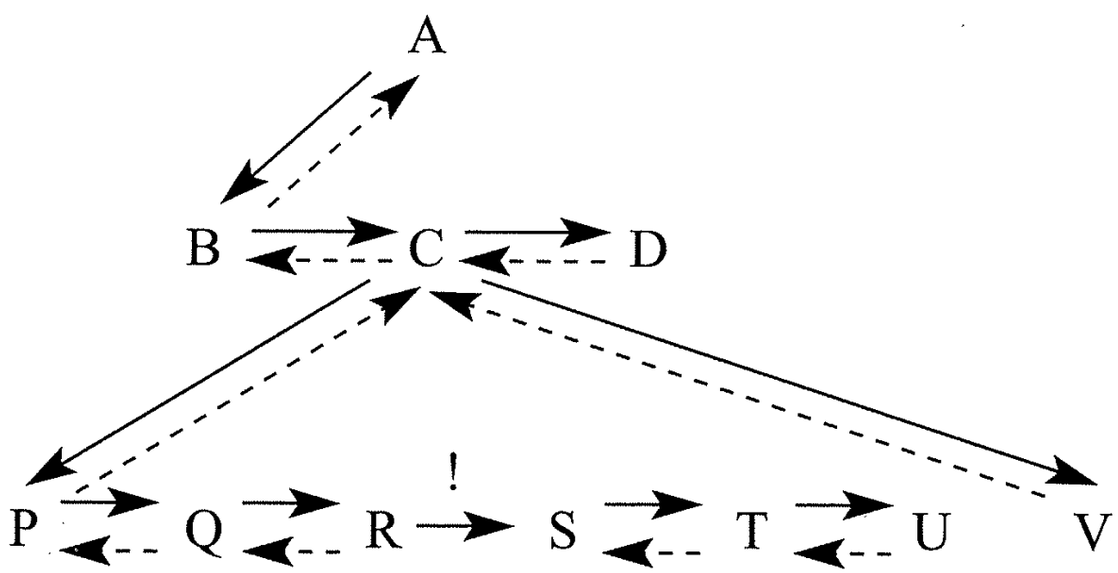


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

The clause containing the cut. →

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

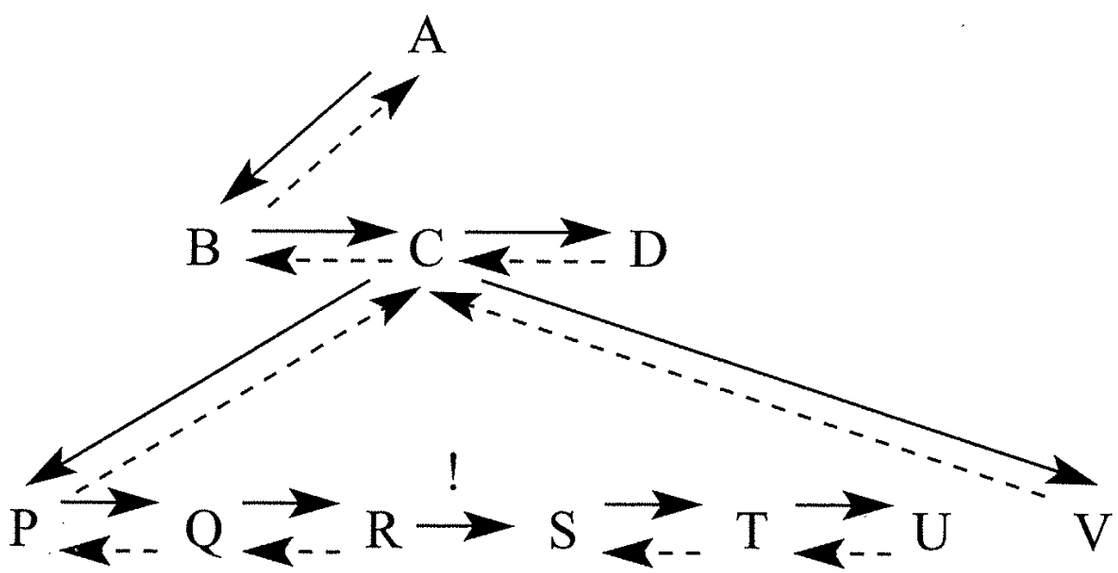


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

Head of the clause containing the cut.

The clause containing the cut.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

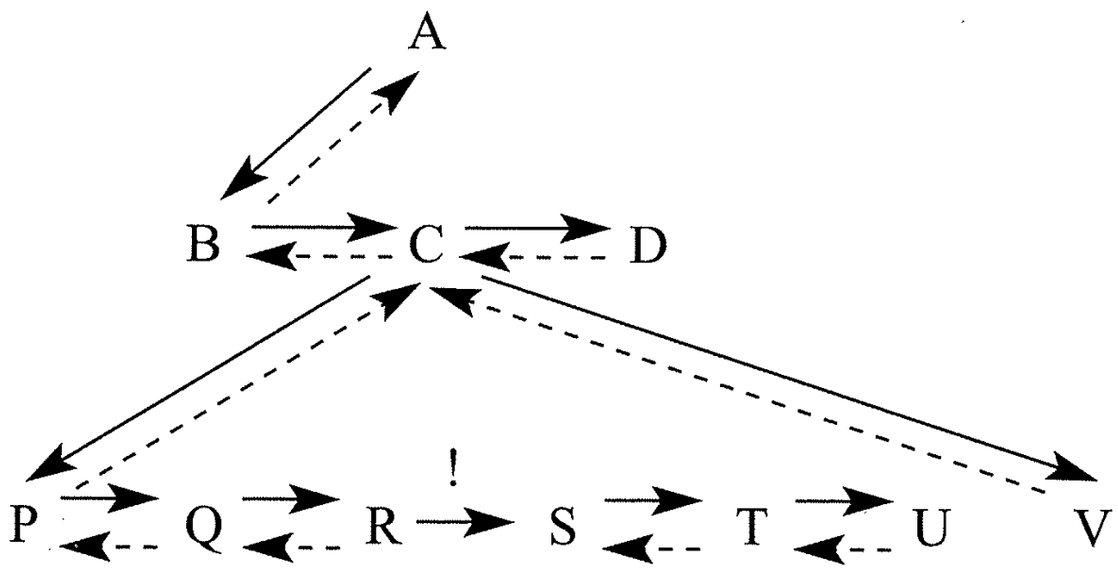


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

Head of the clause containing the cut.

The clause containing the cut.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

C is the parent goal.

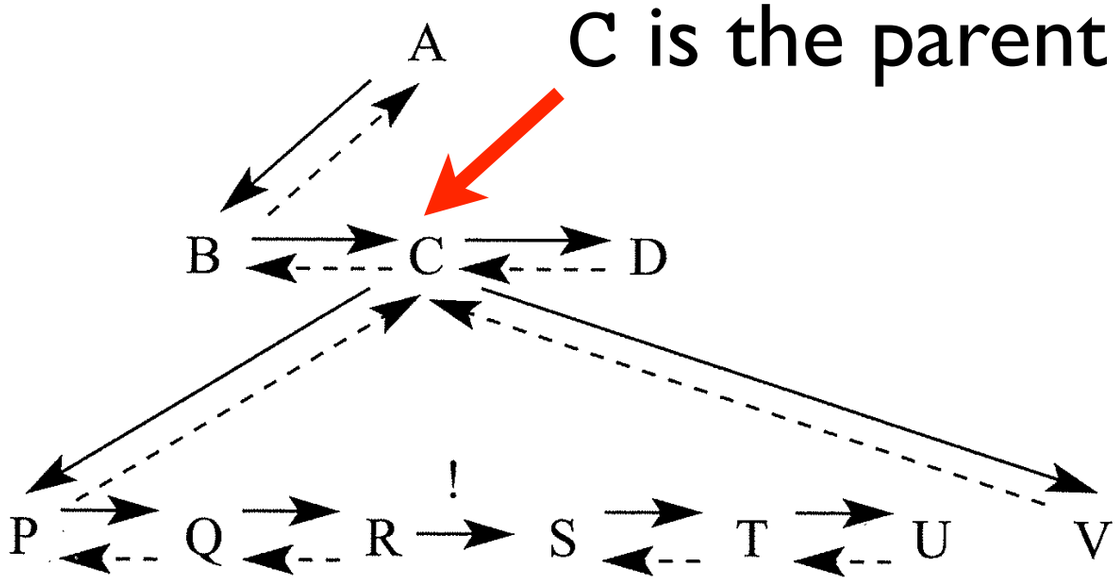


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

Cuts always succeed.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

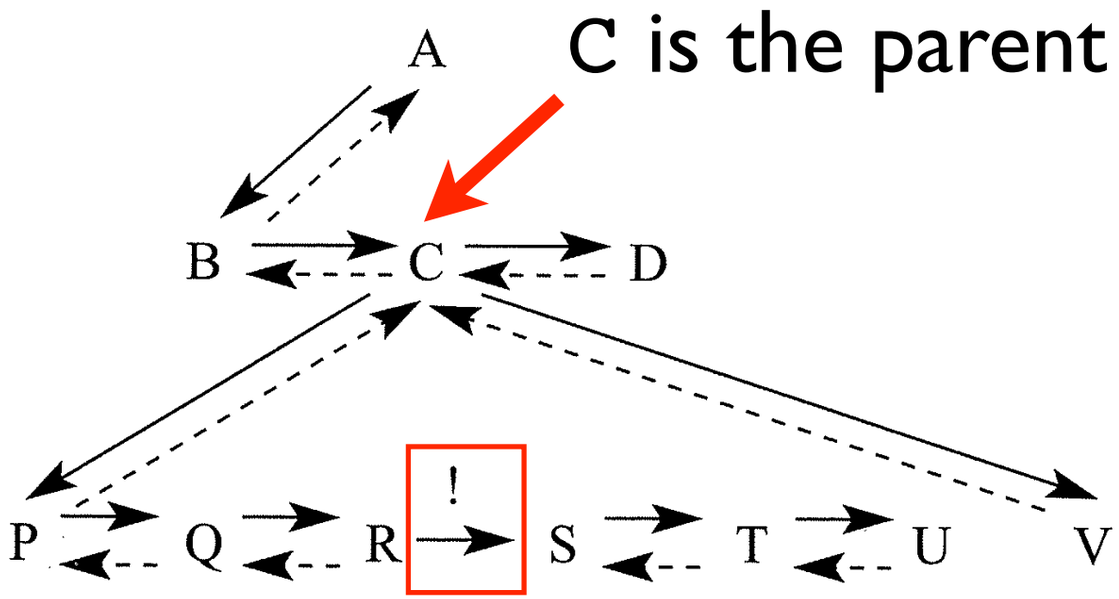


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

Commits the system to these choices.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

C is the parent goal.

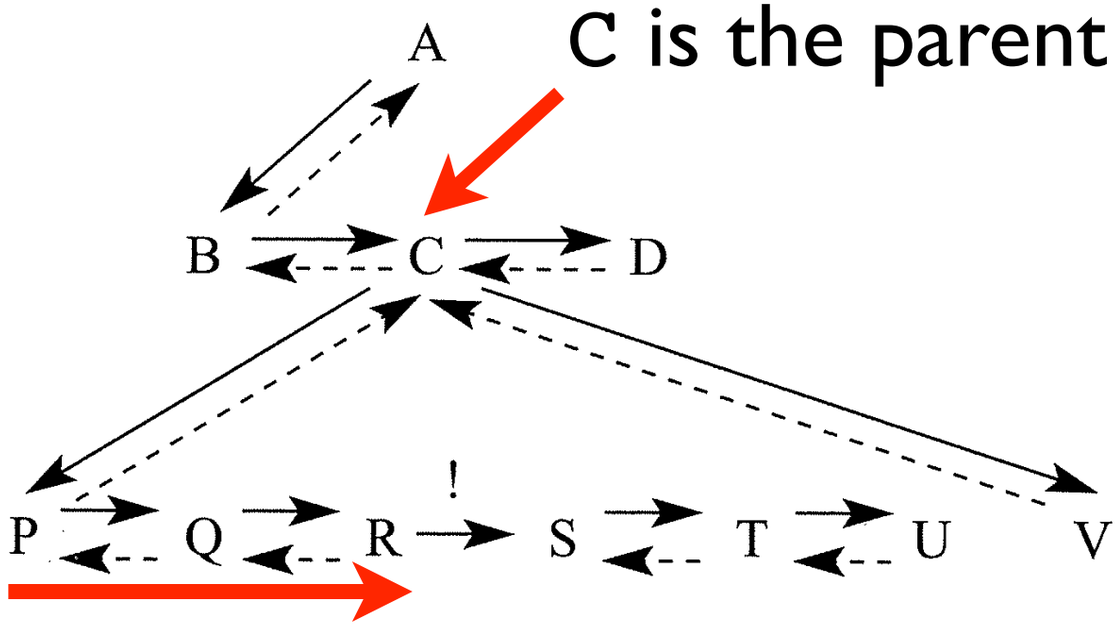


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

Cut fails on backtracking.
Success only possible through S, T, U.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

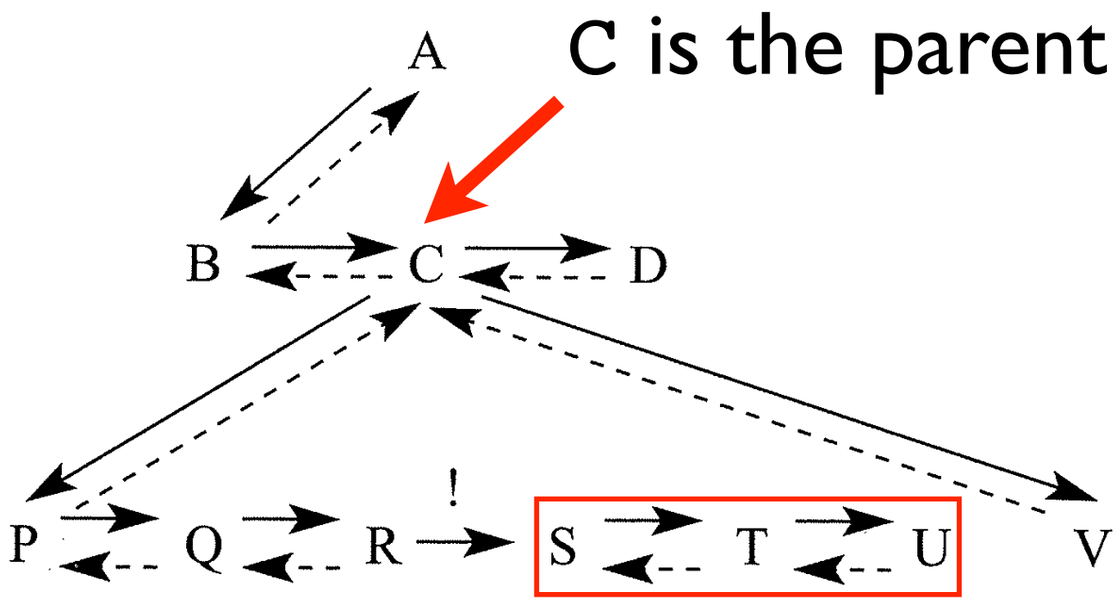


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

Discarded.

```
C :- P, Q, R, !, S, T, U.
C :- V.
```

C is the parent goal.

```
A :- B, C, D.

?- A.
```

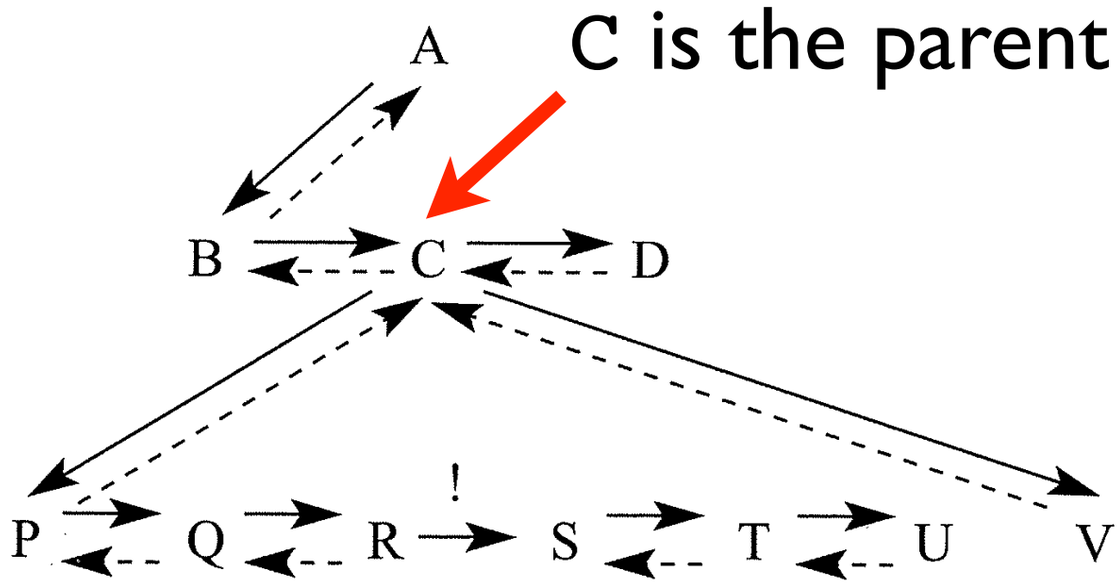


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

The meaning of the cut mechanism

Let us call the 'parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the 'parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded.

If Q fails, the query can succeed through V.

If T fails, the query cannot reach V.

```
C :- P, Q, R, !, S, T, U.

C :- V.

A :- B, C, D.

?- A.
```

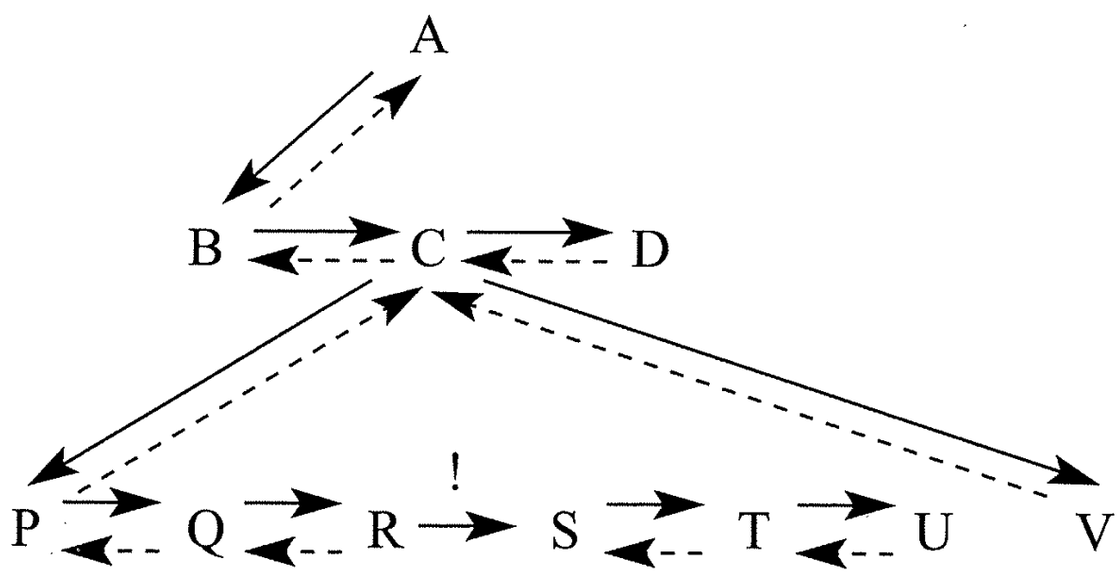


Figure 5.3 The effect of the cut on the execution. Starting with A, the solid arrows indicate the sequence of calls; the dashed arrows indicate backtracking. There is 'one way traffic' between R and S.

Generalized De Morgan

Suppose you have a database of facts that includes all the good_food restaurants and all the expensive restaurants, if any.

The database represents a poor neighborhood if there are no expensive restaurants.

Prolog can determine the truth of the LHS, but it cannot determine the truth of the RHS.

$$\neg(\exists R \mid : \text{expensive}(R)) \equiv (\forall R \mid : \neg\text{expensive}(R))$$

Generalized De Morgan

Suppose you have a database of facts that includes all the good_food restaurants and all the expensive restaurants, if any.

The database represents a poor neighborhood if there are no expensive restaurants.

Prolog can determine the truth of the LHS, but it cannot determine the truth of the RHS.

$$\neg(\exists R \mid : \text{expensive}(R)) \equiv (\forall R \mid : \neg\text{expensive}(R))$$

Even if R is not instantiated, Prolog can search for the existence of an expensive R with backtracking.

If it succeeds in finding fact
`expensive( geoffreys).`
it has proved that the neighborhood is not poor.

If it fails to find any expensive facts it has proved that the neighborhood is poor.

Generalized De Morgan

Suppose you have a database of facts that includes all the good_food restaurants and all the expensive restaurants, if any.

The database represents a poor neighborhood if there are no expensive restaurants.

Prolog can determine the truth of the LHS, but it cannot determine the truth of the RHS.

$$\neg(\exists R \mid : \text{expensive}(R)) \equiv (\forall R \mid : \neg\text{expensive}(R))$$

Even if R is not instantiated, Prolog can search for the existence of an expensive R with backtracking.

If it succeeds in finding fact
`expensive( geoffreys).`
it has proved that the neighborhood is not poor.

If it fails to find any expensive facts it has proved that the neighborhood is poor.

If R is not instantiated, Prolog cannot search over all the restaurants.

It does not have a set of atoms to use in its search.