```
F1:   spec(mach1,ibmpc,320).
F2:   spec(mach2,mac,1000).
F3:   spec(mach3,ibmpc,640).
F4:   runs(ibmpc,spreadsheet,500).
F5:   runs(ibmpc,basic,128).
F6:   runs(ibmpc,pascal,256).
F7:   runs(mac,basic,200).
F8:   runs(mac,smalltalk,1000).
F9:   access(sue,mach1).
F10:  access(jerry,mach3).
F11:  access(sam,mach1).
F12:  access(sam,mach2).
F13:  written_in(spreadsheet,pascal).
R1:   can_use(P,SW) :- access(P,M),
                       can_run(M,SW).
R2:   can_run(M,SW) :- spec(M,HW,Mem1),
                         runs(HW,SW,Mem2),
                         Mem1>=Mem2.
R3:   can_run(M,SW) :- written_in(SW,L),can_run(M,L).
```

If we run the corresponding program with the goal

```
?-can_use(X,spreadsheet).
```

we will obtain the following output:

```
X = sue
X = jerry
X = jerry
X = sam
```

Notice that jerry is derived twice, indicating that the program is doing some unnecessary work. Let's see why this happened. The object `jerry` will satisfy `can_use` once because `jerry` has access to `mach3`, which can run a spreadsheet directly. But the goal can also be derived from the second rule for `can_run` since Pascal can run on `mach3` as well and the spreadsheet is written in Pascal.

The cut is considered to be a subgoal that is universally satisfied but that cannot be backed up over. It then stands as a one-way gate for the subgoal generation process, which, once passed, cannot be backed through.

In the previous example, the first rule for `can_run` could be rewritten with the cut as follows:

```
can_run(MACH,SW) :- spec(MACH,HW,MEM1),
                    runs(HW,SW,MEM2),
                    MEM1 >= MEM2,
                    !.
```

The cut is signified by an exclamation point (`!`).

After the first three subgoals are satisfied, the cut is then satisfied as well, completing the derivation of `can_run`. After the entire derivation is completed, the cut then prevents the backing up process from trying other alternatives for `can_run`. Thus unnecessary checking is avoided, since it is irrelevant how many ways a computer `can_run` a software product.

Now the goal

```
?-can_use(X,spreadsheet).
```

will find only one derivation for each of the three people, and the derivation process is displayed in Figure 11.9. The derivation process, when backing up to a cut, will automatically produce a failure for all subgoals

FIGURE 11.9 Derivation of a goal in Prolog using cut

```
1. ?-can_use(X,spreadsheet).                                    R1/SW:=spreadsheet.
2. ?-access(X,M),can_run(M,spreadsheet).                        F9/M:=mach1,X:=sue
3. ?-can_run(mach1,spreadsheet).                                R2
4. ?-spec(mach1,HW,M1),runs(HW,spreadsheet,M2),M1>=M2,!.        F1/HW:=ibmpc,M1:=320
5. ?-runs(ibmpc,spreadsheet,M2),320>=M2,!.                      F4/M2:=500
6. ?-320>=500,!.                                                Fails
5. ?-runs(ibmpc,spreadsheet,M2),320>=M2,!.                      Fails
4. ?-spec(mach1,HW,M1),runs(HW,spreadsheet,M2),M1>=M2,!.        Fails
3. ?-can_run(mach1,spreadsheet).                                R3
4. ?-written_in(spreadsheet,L),can_run(mach1,L).                F13/L:=pascal
5. ?-can_run(mach1,pascal).                                     R2
6. ?-spec(mach1,HW,M1),runs(HW,pascal,M2),M1>=M2,!.             F1/HW:=ibmpc,M1:=320
7. ?-runs(ibmpc,pascal,M2),320>=M2,!.                           F6/M2:=256
8. ?-320>=256,!.                                                satisfied
9. ?-! Skip backup because of cut - 5. completed                satisfied  X=sue
4. ?-written_in(spreadsheet,L),can_run(mach1,L)                 Fails
3. ?-can_run(mach1,spreadsheet).                                Fails
2. ?-access(X,M),can_run(M,spreadsheet).                        F10/X:=jerry,M:=mach3
3. ?-can_run(mach3,spreadsheet).                                R2
4. ?-spec(mach3,HW,M1),runs(HW,spreadsheet,M2),M1>=M2,!.        F3/M1:=640,HW=ibmpc
5. ?-runs(ibmpc,spreadsheet,M2),640>=M2,!.                      F4/M2:=500
6. ?-640>=500,!.                                                satisfied
7. ?-! Skip backup because of cut - 3. completed                satisfied  X=jerry
2. ?-access(X,M),can_run(M,spreadsheet).                        F11/X:=sam,M:=mach1
3. ?-can_run(mach1,spreadsheet).                                R2
4. ?-spec(mach1,HW,M1),runs(HW,spreadsheet,M2),M1>=M2,!.        F1/HW:=ibmpc,M1:=320
5. ?-runs(ibmpc,spreadsheet,M2),320>=M2,!.                      F4/M2:=500
6. ?-320>=500,!.                                                Fails
5. ?-runs(ibmpc,spreadsheet,M2),320>=M2,!.                      Fails
4. ?-spec(mach1,HW,M1),runs(HW,spreadsheet,M2),M1>=M2,!.        Fails
3. ?-can_run(mach1,spreadsheet).                                R3
4. ?-written_in(spreadsheet,L),can_run(mach1,L).                F13/L:=pascal
5. ?-can_run(mach1,pascal).                                     R2
6. ?-spec(mach1,HW,M1),runs(HW,pascal,M2),M1>=M2,!.             F1/HW:=ibmpc,M1:=320
7. ?-runs(ibmpc,pascal,M2),320>=M2,!.                           F6/M2:=256
8. ?-320>=256,!.                                                satisfied
9. ?-! Skip backup because of cut - 5. completed                satisfied  X=sam
4. ?-written_in(spreadsheet,L),can_run(mach1,L).                Fails
3. ?-can_run(mach1,spreadsheet).                                Fails
2. ?-access(X,M),can_run(M,spreadsheet).                        F12/X:=sam,M:=mach2
3. ?-can_run(mach2,spreadsheet).                                R2
4. ?-spec(mach2,HW,M1),runs(HW,spreadsheet,M2),M1>=M2,!.        F2/HW:=mac,M1:=1000
5. ?-runs(mac,spreadsheet,M2),1000>=M2,!.                       Fails
4. ?-spec(mach2,HW,M1),runs(HW,spreadsheet,M2),M1>=M2,!.        Fails
3. ?-can_run(mach2,spreadsheet).                                R3
4. ?-written_in(spreadsheet,L),can_run(mach2,L).                F13/L:=pascal
5. ?-can_run(mach2,pascal).                                     R2
6. ?-spec(mach2,HW,M1),runs(HW,pascal,M2),M1>=M2,!.             F2/HW:=mac,M1:=1000
7. ?-runs(mac,pascal,M2),M1>=M2,!.                              Fails
6. ?-spec(mach2,HW,M1),runs(HW,pascal,M2),M1>=M2,!.             Fails
5. ?-can_run(mach2,pascal).                                     R3
6. ?-written_in(pascal,L),can_run(mach2,L).                     Fails
5. ?-can_run(mach2,pascal).                                     Fails
4. ?-written_in(spreadsheet,L),can_run(mach2,L).                Fails
3. ?-can_run(mach2,spreadsheet).                                Fails
2. ?-access(X,M),can_run(M,spreadsheet).                        Fails
1. ?-can_use(X,spreadsheet).
```