Figure 25.1 (a) Prolog notation. (b) The supervisory tree.

(a)

```
        Facts
supervise(franklin,john).
supervise(franklin,ramesh).
supervise(franklin,joyce).
supervise(jennifer,alicia).
supervise(jennifer,ahmad).
supervise(james,franklin).
supervise(james,jennifer).
      ...

        Rules
superior(X,Y) :- supervise(X,Y).
superior(X,Y) :- supervise(X,Z), superior(Z,Y).
subordinate(X,Y) :- superior(Y,X).

        Queries
superior(james,Y)?
superior(james,joyce)?
```

(b)

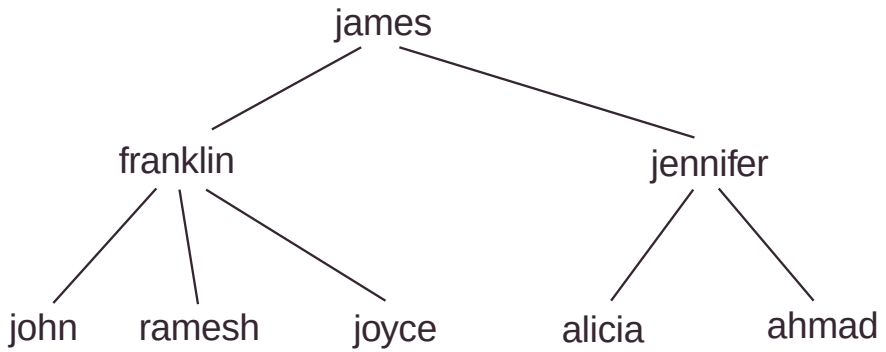

Figure 25.2 Proving a new fact.

1. superior(X,Y) :- supervise(X,Y).	(rule 1)
2. superior(X,Y) :- supervise(X,Z), superior(Z,Y).	(rule 2)
3. supervise(jennifer,ahmad).	(ground axiom, given)
4. supervise(james,jennifer).	(ground axiom, given)
5. superior(jennifer,ahmad).	(apply rule 1 on 3)
6. superior(james,ahmad).	(apply rule 2 on 4 and 5)

Figure 25.3 An interpretation that is a minimal model.

Rules

superior(X,Y) :- supervise(X,Y).
superior(X,Y) :- supervise(X,Z), superior(Z,Y).

Interpretation

Known Facts:

supervise(franklin,john) is **true**.
supervise(franklin,ramesh) is **true**.
supervise(franklin,joyce) is **true**.
supervise(jennifer,alicia) is **true**.
supervise(jennifer,ahmad) is **true**.
supervise(james,franklin) is **true**.
supervise(james,jennifer) is **true**.
supervise(X,Y) is **false** for all other possible (X,Y) combinations.

Derived Facts:

superior(franklin,john) is **true**.
superior(franklin,ramesh) is **true**.
superior(franklin,joyce) is **true**.
superior(jennifer,alicia) is **true**.
superior(jennifer,ahmad) is **true**.
superior(james,franklin) is **true**.
superior(james,jennifer) is **true**.
superior(james,john) is **true**.
superior(james,ramesh) is **true**.
superior(james,joyce) is **true**.
superior(james,alicia) is **true**.
superior(james,ahmad) is **true**.
superior(X,Y) is **false** for all other possible (X,Y) combinations.

Figure 25.4 Top-down evaluation of a query.

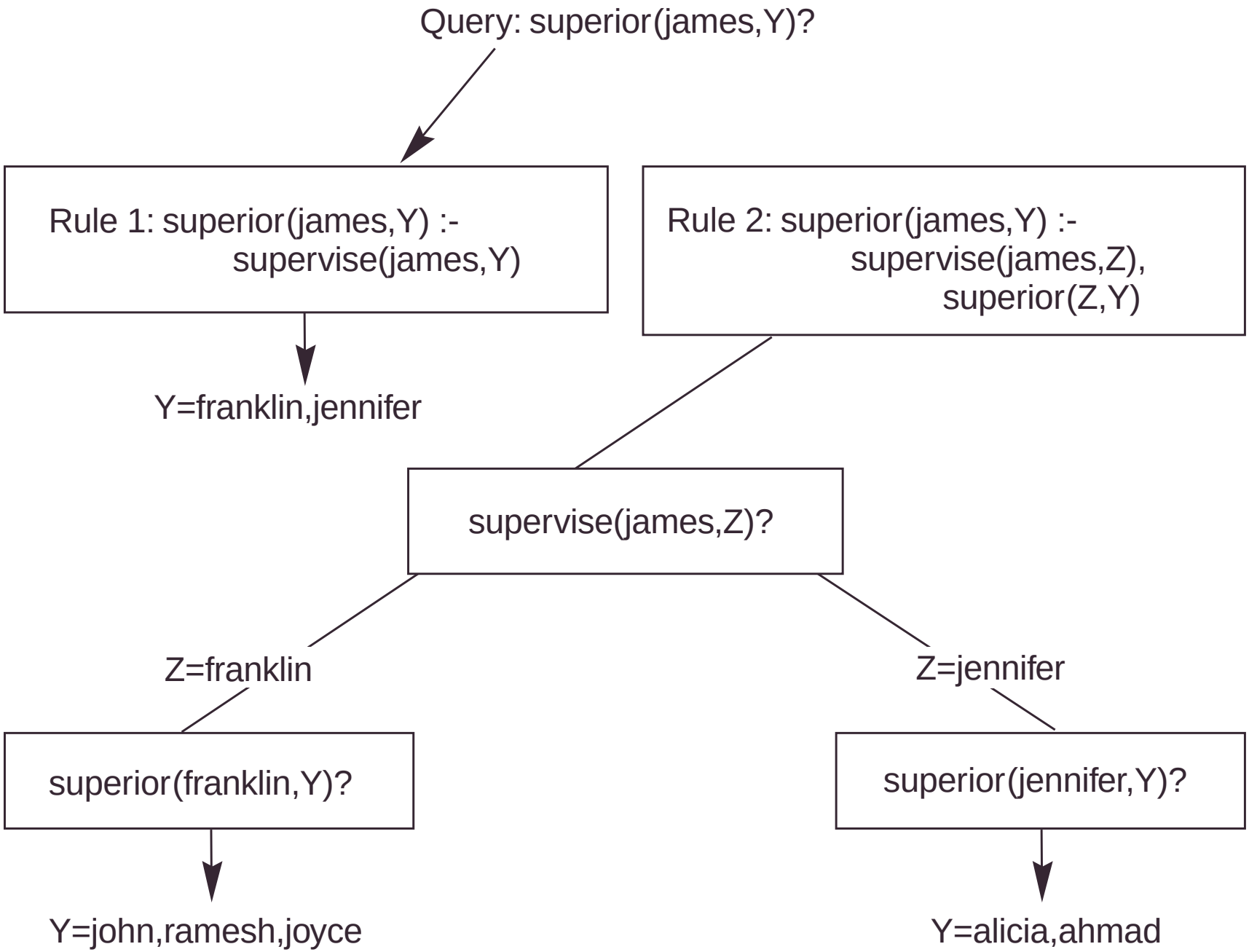

Figure 25.5 Fact predicates for part of the database from Figure 7.6.

```
employee(john).
employee(franklin).
employee(alicia).
employee(jennifer).
employee(ramesh).
employee(joyce).
employee(ahmad).
employee(james).

salary(john,30000).
salary(franklin,40000).
salary(alicia,25000).
salary(jennifer,43000).
salary(ramesh,38000).
salary(joyce,25000).
salary(ahmad,25000).
salary(james,55000).

department(john,research).
department(franklin,research).
department(alicia,administration).
department(jennifer,administration).
department(ramesh,research).
department(joyce,research).
department(ahmad,administration).
department(james,headquarters).

supervise(franklin,john).
supervise(franklin,ramesh).
supervise(franklin,joyce).
supervise(jennifer,alicia).
supervise(jennifer,ahmad).
supervise(james,franklin).
supervise(james,jennifer).

male(john).
male(franklin).
male(ramesh).
male(ahmad).
male(james).

female(alicia).
female(jennifer).
female(joyce).

project(productx).
project(producty).
project(productz).
project(computerization).
project(reorganization).
project(newbenefits).

workson(john,productx,32).
workson(john,producty,8).
workson(ramesh,productz,40).
workson(joyce,productx,20).
workson(joyce,producty,20).
workson(franklin,producty,10).
workson(franklin,productz,10).
workson(franklin,computerization,10).
workson(franklin,reorganization,10).
workson(alicia,newbenefits,30).
workson(alicia,computerization,10).
workson(ahmad,computerization,35).
workson(ahmad,newbenefits,5).
workson(jennifer,newbenefits,20).
workson(jennifer,reorganization,15).
workson(james,reorganization,10).
```

Figure 25.6 Rule-defined predicates.

```
superior(X,Y) :- supervise(X,Y).
superior(X,Y) :- supervise(X,Z), superior(Z,Y).

subordinate(X,Y) :- superior(Y,X).

supervisor(X) :- employee(X), supervise(X,Y).

over_40K_emp(X) :- employee(X), salary(X,Y), Y>=40000.
under_40K_supervisor(X) :- supervisor(X), not(over_40_K_emp(X)).
main_productx_emp(X) :- employee(X), workson(X,productx,Y), Y>=20.
president(X) :- employee(X), not(supervise(Y,X)).
```

Figure 25.7 Predicates for illustrating relational operations.

```
rel_one(A,B,C).
rel_two(D,E,F).
rel_three(G,H,I,J).

select_one_A_eq_c(X,Y,Z) :- rel_one(c,Y,Z).
select_one_B_less_5(X,Y,Z) :- rel_one(X,Y,Z), Y<5.
select_one_A_eq_c_and_B_less_5(X,Y,Z) :- rel_one(c,Y,Z), Y<5.

select_one_A_eq_c_or_B_less_5(X,Y,Z) :- rel_one(c,Y,Z).
select_one_A_eq_c_or_B_less_5(X,Y,Z) :- rel_one(X,Y,Z), Y<5.

project_three_on_G_H(W,X) :- rel_three(W,X,Y,Z).

union_one_two(X,Y,Z) :- rel_one(X,Y,Z).
union_one_two(X,Y,Z) :- rel_two(X,Y,Z).

intersect_one_two(X,Y,Z) :- rel_one(X,Y,Z), rel_two(X,Y,Z).

difference_two_one(X,Y,Z) :- rel_two(X,Y,Z), not(rel_one(X,Y,Z)).

cart_prod_one_three(T,U,V,W,X,Y,Z) :-
                                rel_one(T,U,V), rel_three(W,X,Y,Z).

natural_join_one_three_C_eq_G(U,V,W,X,Y,Z) :-
                                 rel_one(U,V,W), rel_three(W,X,Y,Z).
```

Figure 25.8 Predicate dependency graph for Figures 25.6 and 25.7.

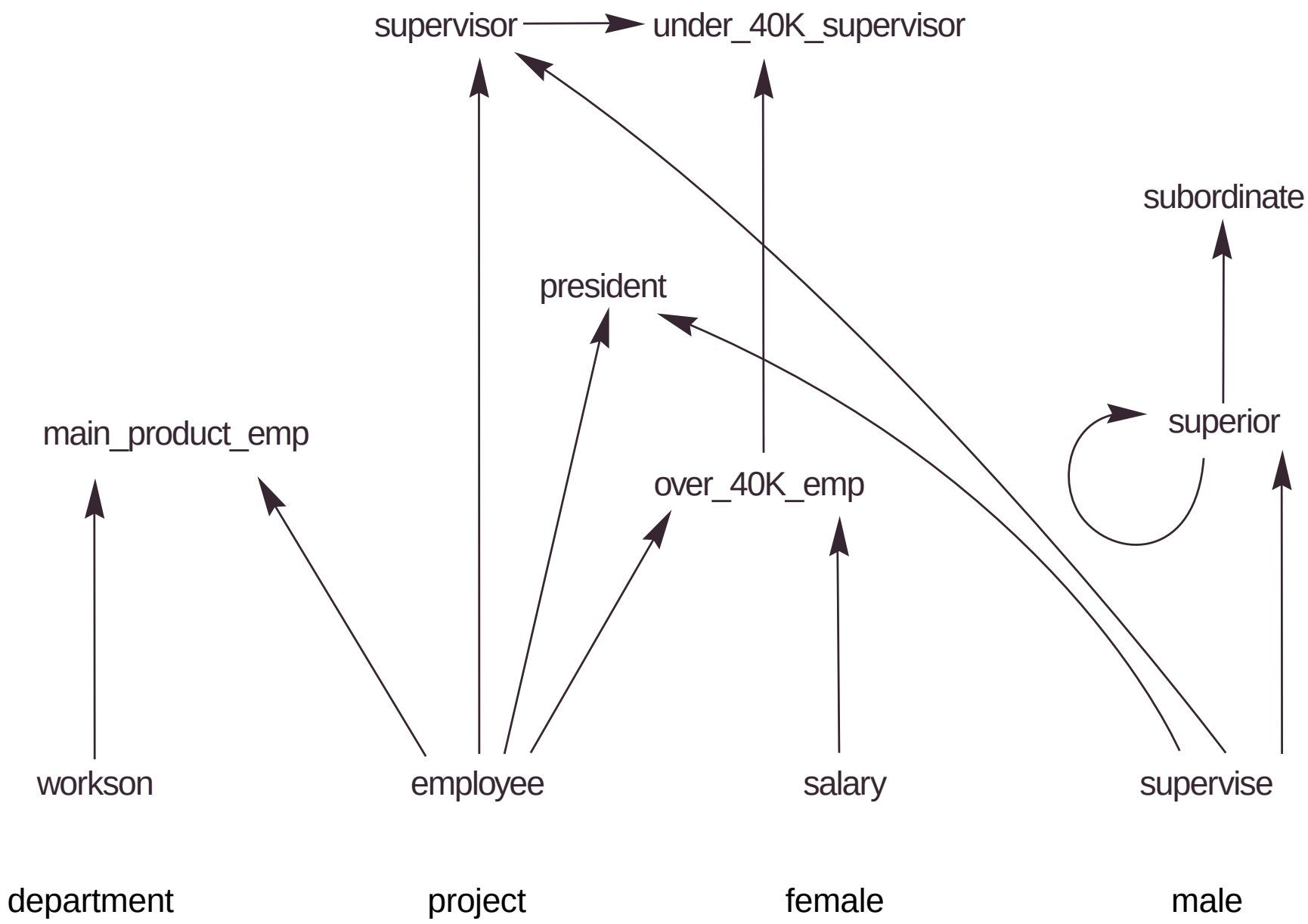

Figure 25.9 Parent tree for example facts.

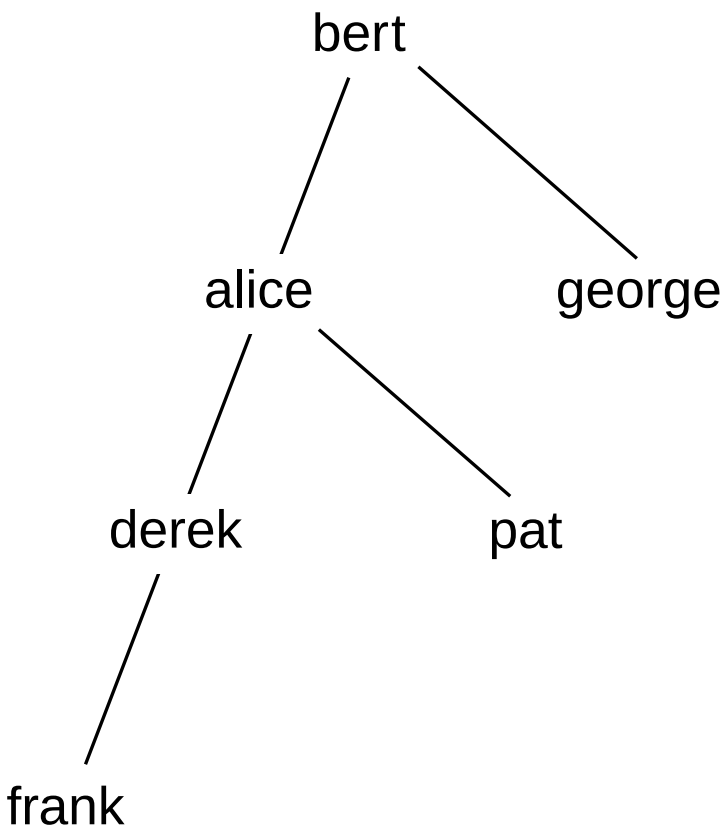

Figure 25.10 NAIL! system architecture.