Matching I

The rule:

```
deriv(U ^C, X, C * U ^L * DU) :-
        number(C), L is C - 1,
        deriv(U, X, DU).

?- deriv(x ^3, x, D).
   D = 1*3*x^2
```

The goal:

- `x ^3` matches `U ^C`
  - $x = U, C = 3$
- `x` matches `X`
- `D` matches `C * U ^L * DU`

Matching II

```
deriv(U+V, X, DU + DV) :-
    deriv(U, X, DU),
    deriv(V, X, DV).

?- deriv(x^3 + x^2 + 1, x, D).
    D = 1*3*x^2+1*2*x^1+0
```

- `x ^3 + x^2 + 1` matches `U + V`
  - `x ^3 + x^2` is bound to `U`
  - `1` is bound to `V`

Matching III

Can two terms A and F be "made identical," by assigning values to their variables?

Two terms A and F match if

1. they are identical atoms
2. one or both are uninstantiated variables
3. they are terms $A = f_A(a_1, \cdots, a_n)$ and $F = f_F(f_1, \cdots, f_m)$, and
   (a) the arities are the same ($n = m$)
   (b) the functors are the same ($f_A = f_F$)
   (c) the arguments match ($a_i \equiv f_i$)

Matching – Example I (a)

A	F	$A \equiv F$	variable subst.
a	a	yes	
a	b	no	
sin(X)	sin(a)	yes	$\theta = \{\text{X=a}\}$
sin(a)	sin(X)	yes	$\theta = \{\text{X=a}\}$
cos(X)	sin(a)	no	
sin(X)	sin(cos(a))	yes	$\theta = \{\text{X=cos(a)}\}$

Matching – Example I (b)

A	F	$A \equiv F$	variable subst.
likes(c, X)	likes(a, X)	no	
likes(c, X)	likes(c, Y)	yes	θ = {X=Y}
likes(X, X)	likes(c, Y)	yes	θ = {X=c, X=Y}
likes(X, X)	likes(c, _)	yes	θ = {X=c, X=_47}
likes(c, a(X))	likes(V, Z)	yes	θ = {V=c,Z=a(X)}
likes(X, a(X))	likes(c, Z)	yes	θ = {X=c,Z=a(X)}

Matching II

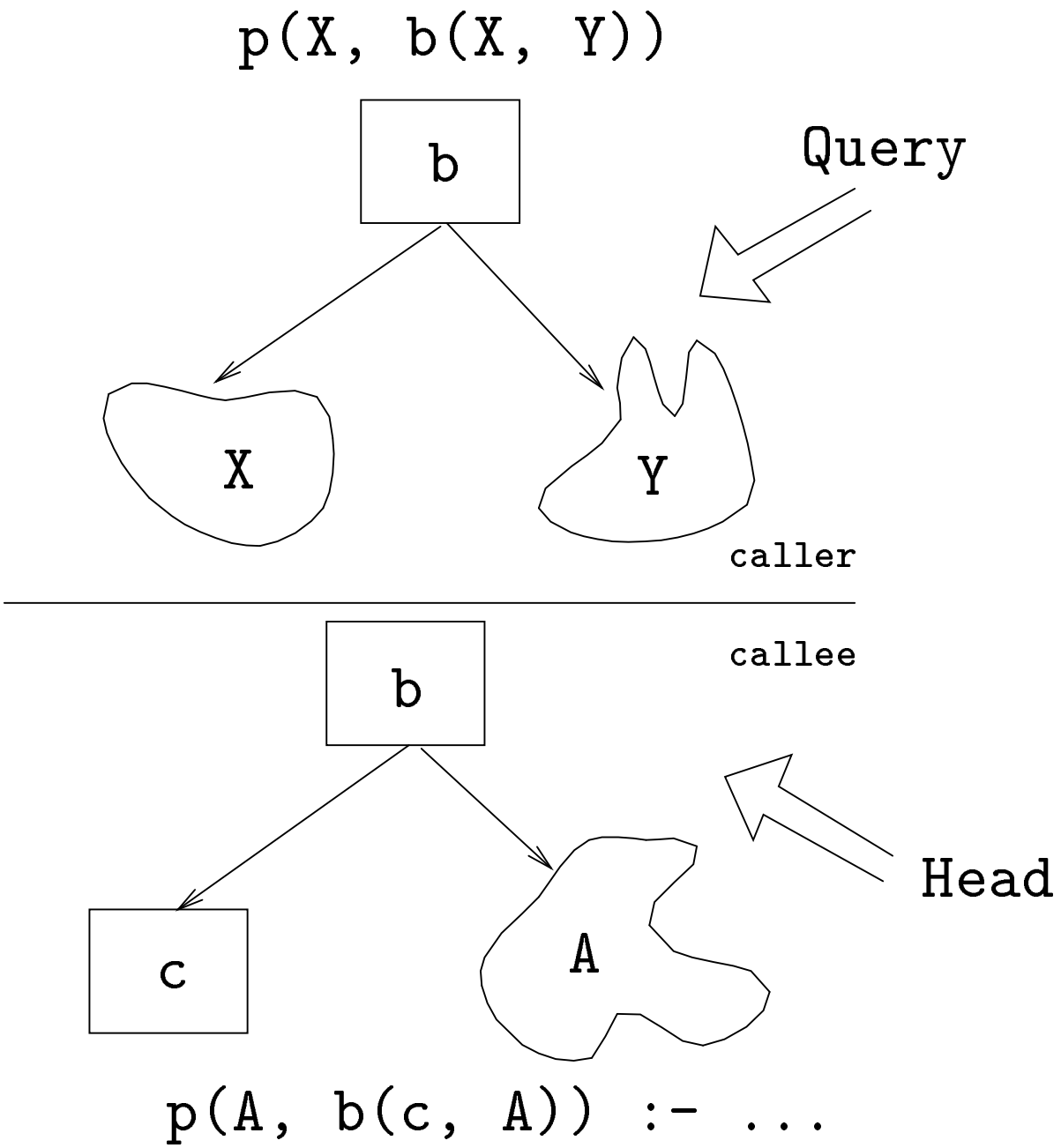
Consequences of Prolog Matching:

- An uninstantiated variable will match any object.
- An integer or atom will match only itself.
- When two uninstantiated variables match, they *share*:
  - When one is instantiated, so is the other (with the same value).
- Backtracking undoes all variable bindings.

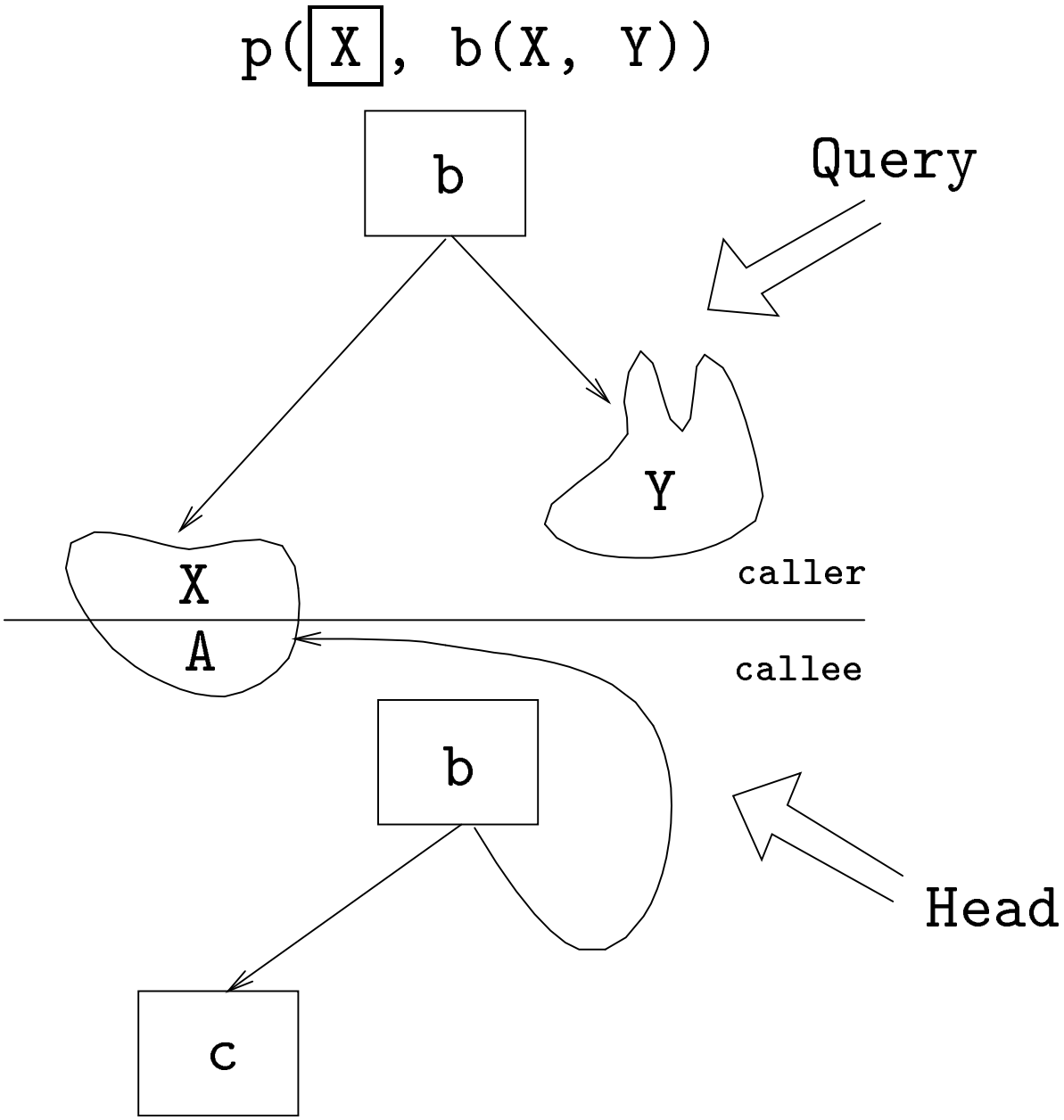
Matching Algorithm

```
FUNC Unify (A, F: term) :  BOOL;
BEGIN
   IF Is_Var(F) THEN
      Instantiate F to A
   ELSIF Is_Var(A) THEN
      Instantiate A to F
   ELSIF Arity(F) ≠ Arity(A) THEN
      RETURN FALSE
   ELSIF Functor(F) ≠ Functor(A) THEN
      RETURN FALSE
   ELSE
      FOR each argument i DO
         IF NOT Unify(A(i), F(i)) THEN
            RETURN FALSE
         ENDIF
      ENDFOR
   ENDIF
   RETURN TRUE;
END
```

Matching – Example II (a)



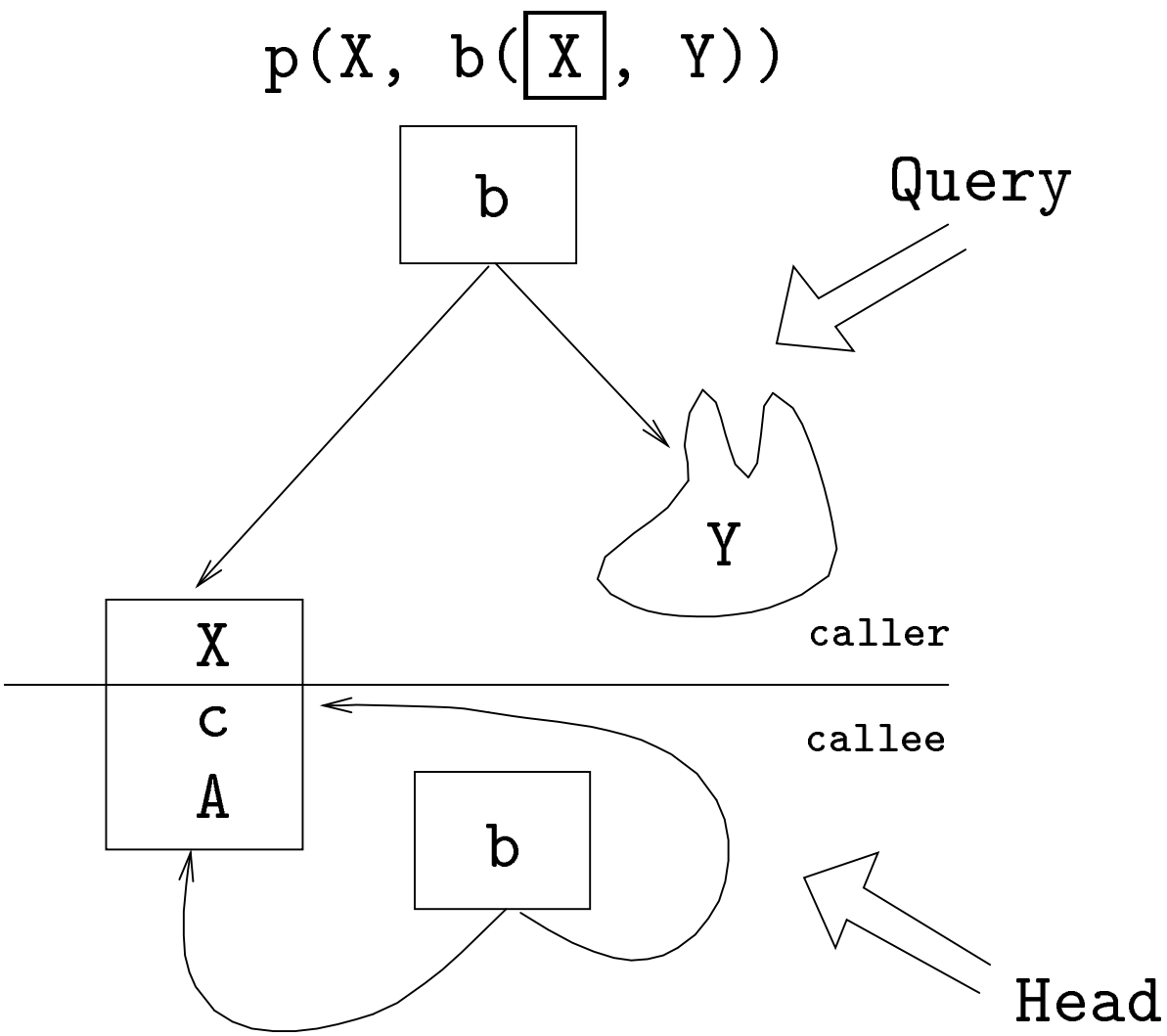
Matching – Example II (b)



p(A , b(c, A)) :- ...

$$\theta = \{A = X\}$$

Matching – Example II (c)

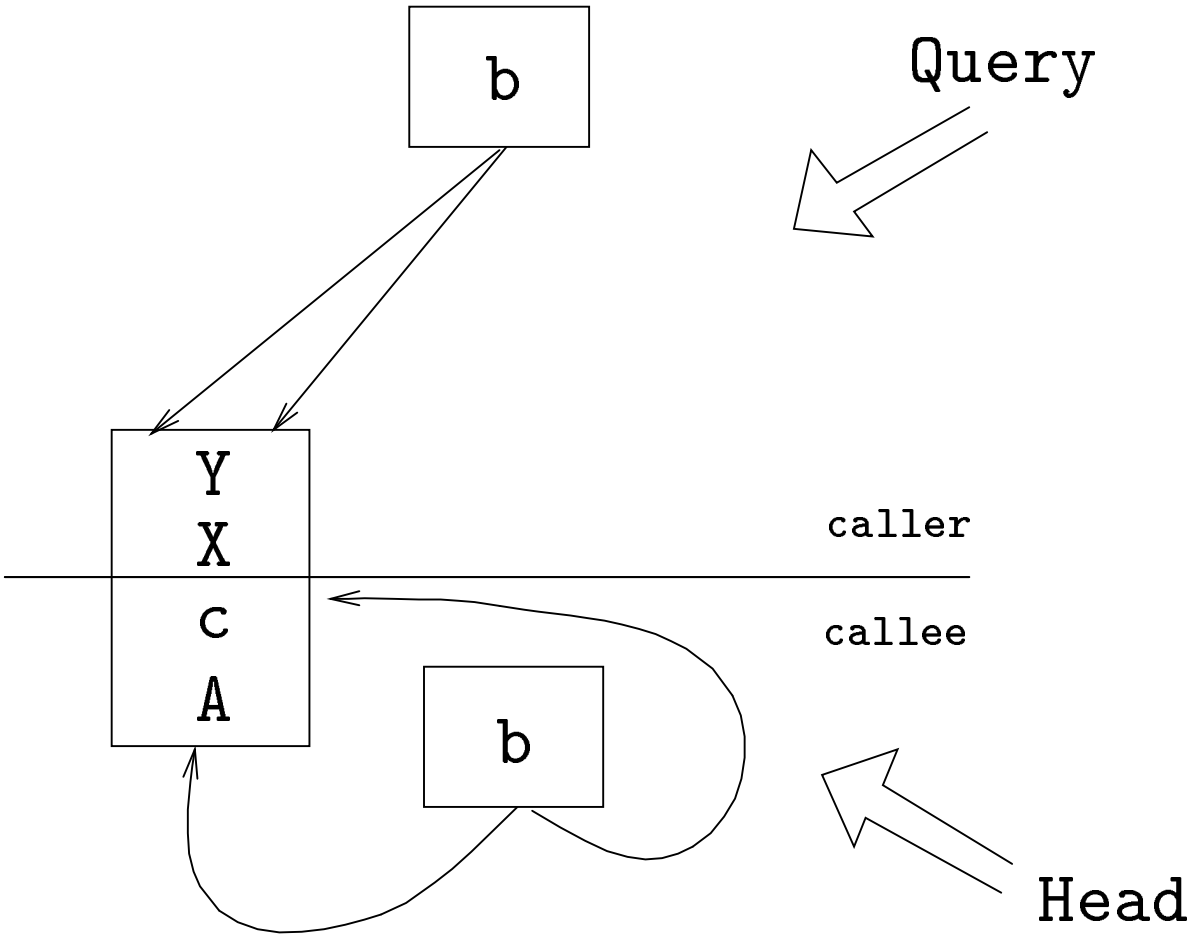


p(A, b(c, A)) :- ...

$$\theta = \{A = X, X = c\}$$

Matching – Example II (d)

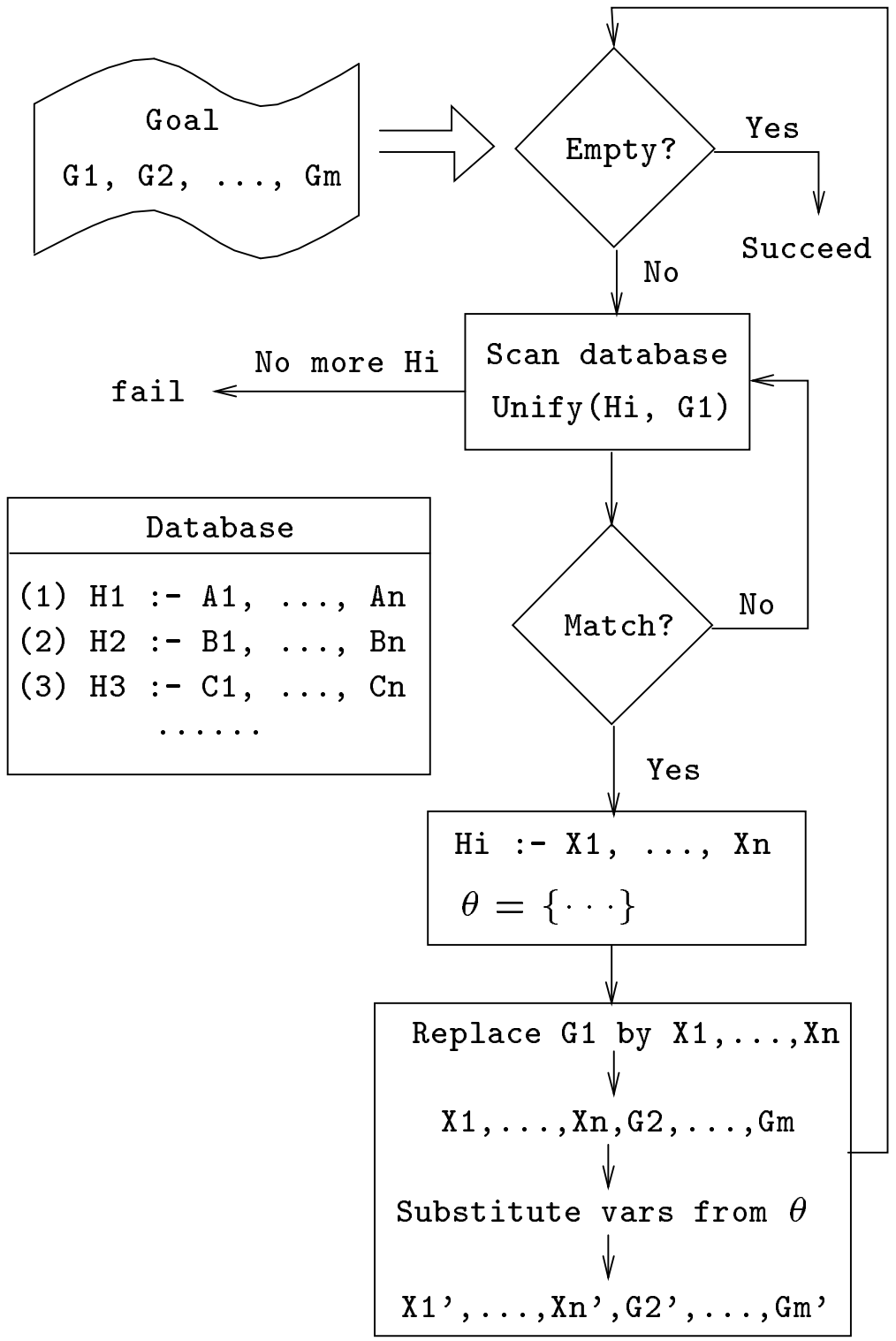
p(X, b(X, Y))



p(A, b(c, A)) :- ...

$$\theta = \{A = X, X = c, A = Y\}$$

Executing Prolog I



Executing Prolog II

FUNC Execute ($G = G_1, G_2, \cdots, G_m$; Result);
  IF Is_Empty(G) **THEN** Result := Yes
  ELSE Result := No; i := 1;
    WHILE Result=No & $i \leq$ NoOfClauses **DO**
      Clause := H_i :- $B_1, \cdots, B_n$;
      IF Unify(G_1, Clause, θ) **THEN**
        G' := substitute($B_1, \cdots, B_n, G_2, \cdots, G_m$, θ);
        Execute(G', Result);
      ENDIF; i := $i + 1$;
    ENDDO ENDIF

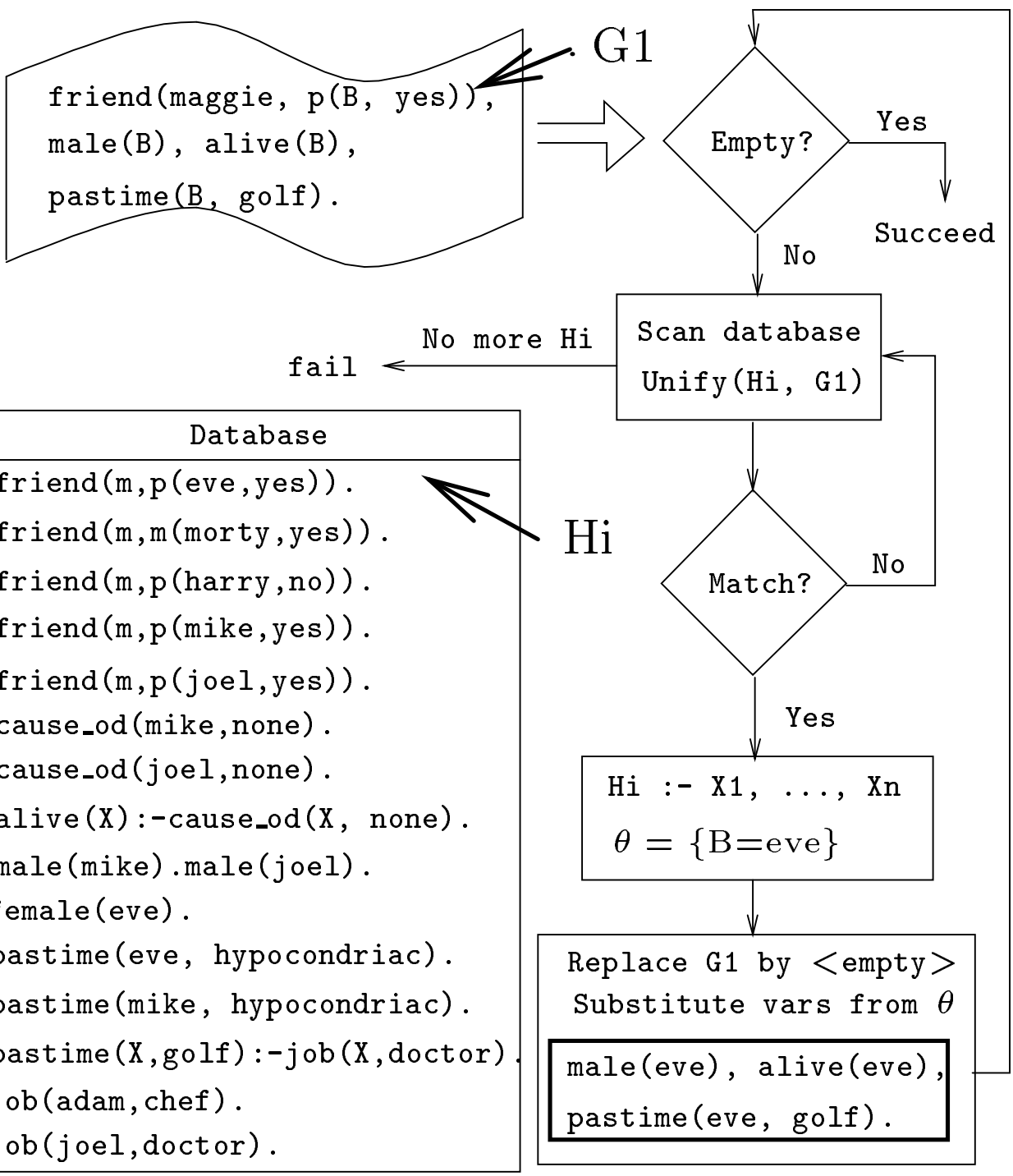
Execution Example I

```
% From the Northern Exposure FAQ
% friend(of, kind(name, regular)).
friend(maggie, person(eve, yes)).
friend(maggie, moose(morty, yes)).
friend(maggie, person(harry, no)).
friend(maggie, person(bruce, no)).
friend(maggie, person(glenn, no)).
friend(maggie, person(dave, no)).
friend(maggie, person(rick, no)).
friend(maggie, person(mike, yes)).
friend(maggie, person(joel, yes)).

cause_of_death(morty, copper_deficiency).
cause_of_death(harry, potato_salad).
cause_of_death(bruce, fishing_accident).
cause_of_death(glenn, missile).
cause_of_death(dave, hypothermia).
cause_of_death(rick, hit_by_satellite).
cause_of_death(mike, none_yet).
cause_of_death(joel, none_yet).
```

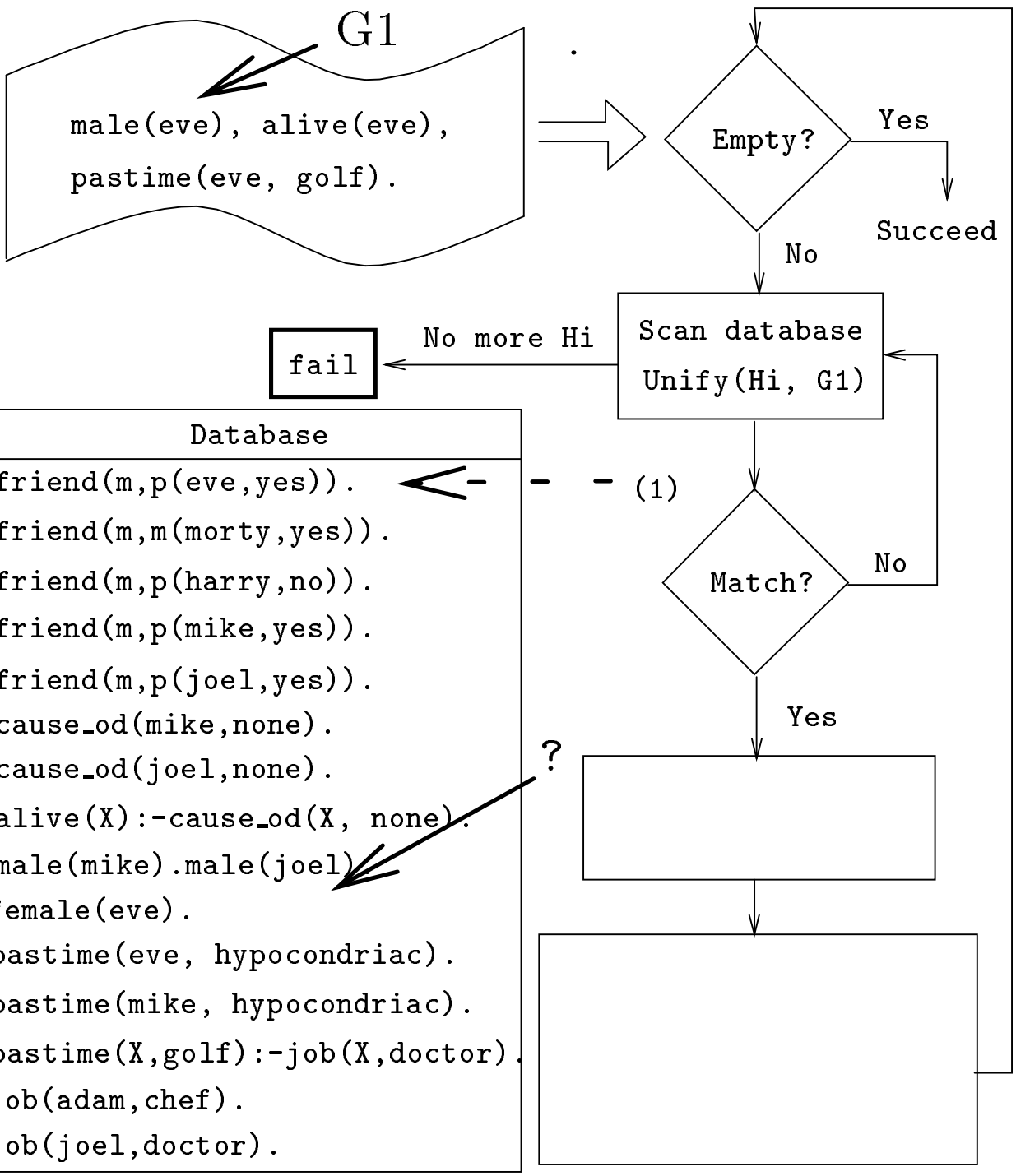
Execution Example II

```
alive(X) :- cause_of_death(X, none_yet).

male(morty).  male(harry).  male(bruce).
male(glenn).  male(dave).  male(rick).
male(mike).  male(joel).  female(eve).

pastime(eve, hypochondria).
pastime(mike, hypochondria).
pastime(X, golf) :- job(X,doctor).

job(mike, lawyer).  job(adam, chef).
job(maggie, pilot).  job(joel, doctor).

?- friend(maggie, person(B, yes)),
      male(B),
      alive(B),
      pastime(B, golf).
```

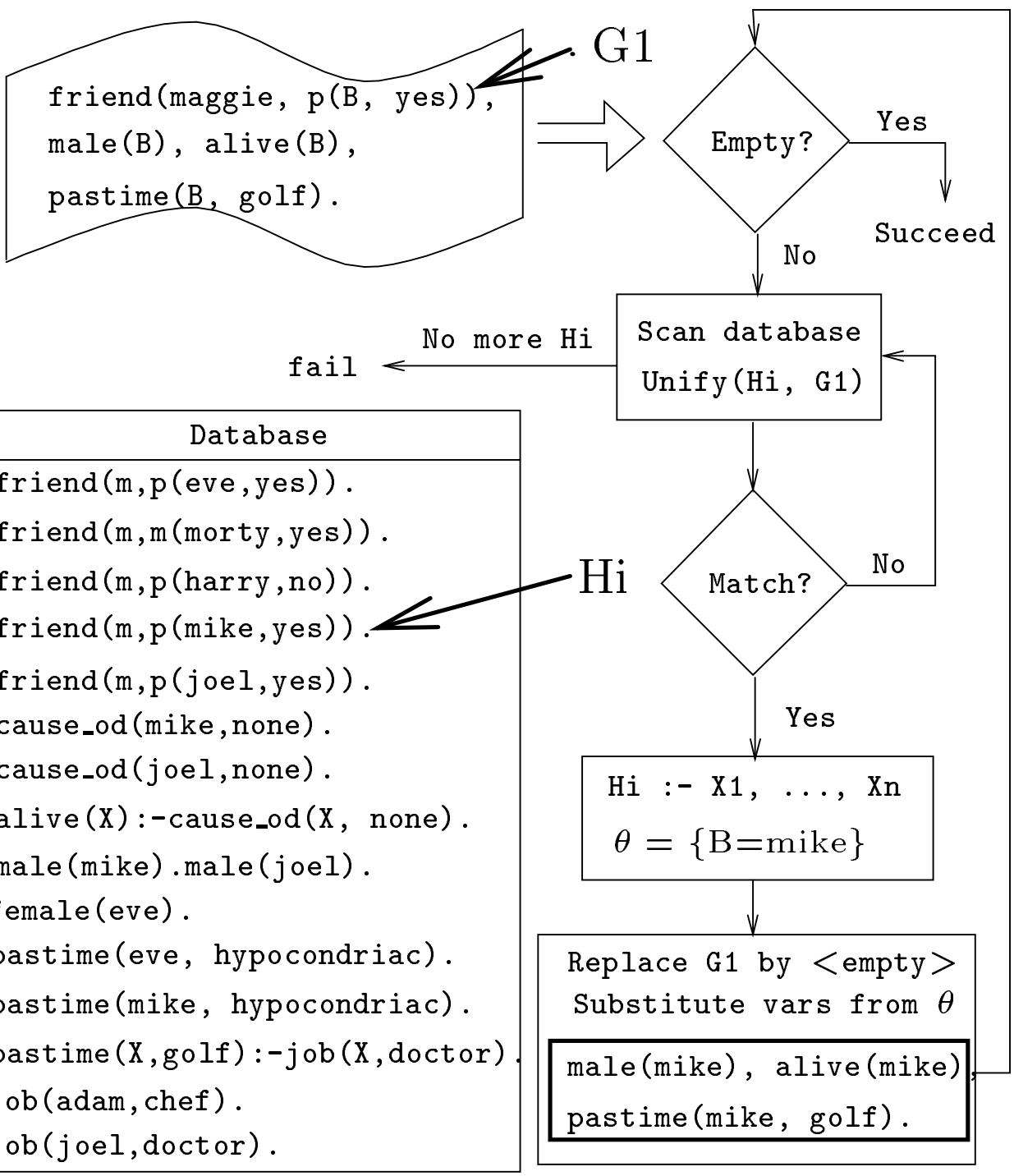
Execution Example III



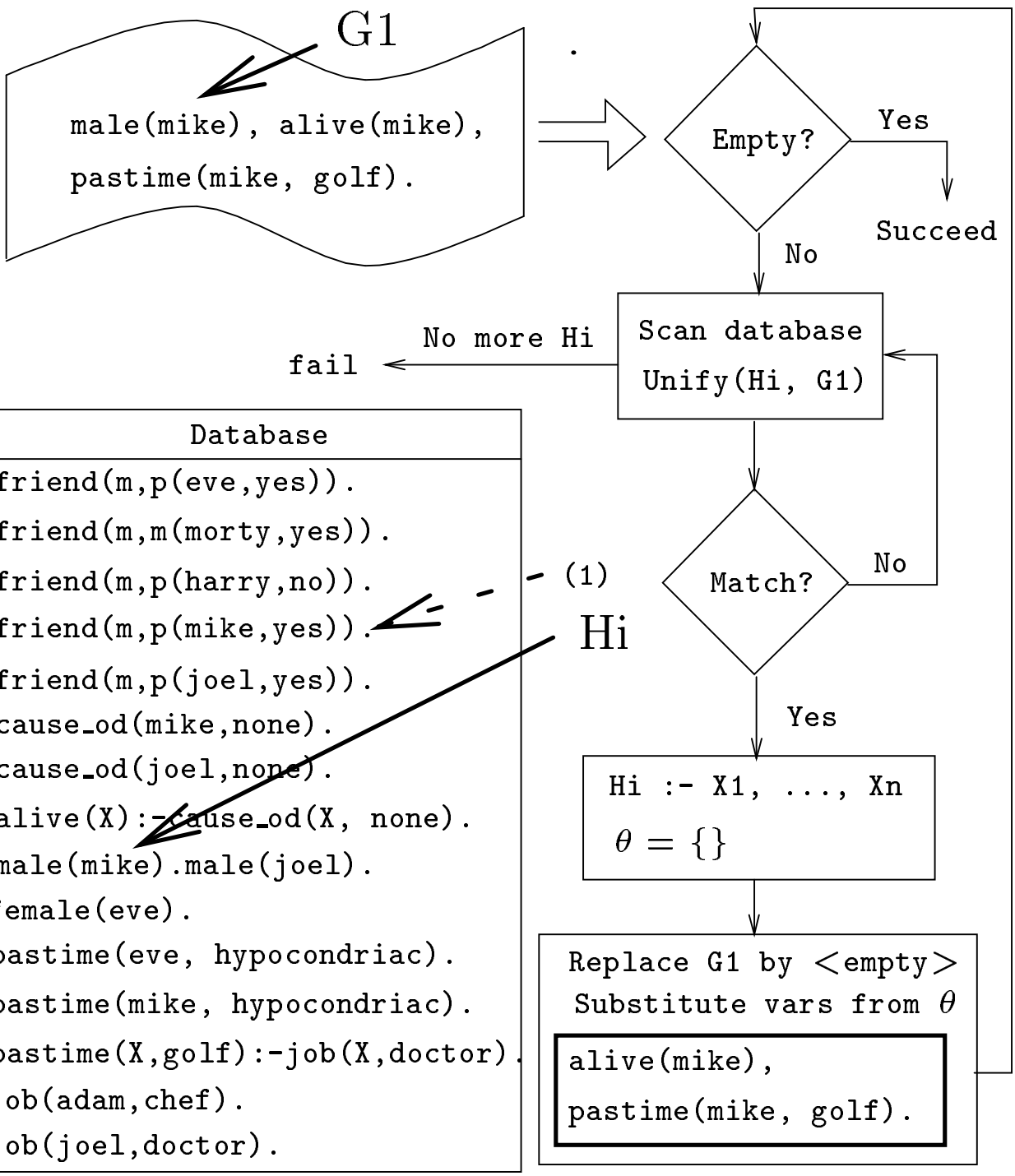
Execution Example IV



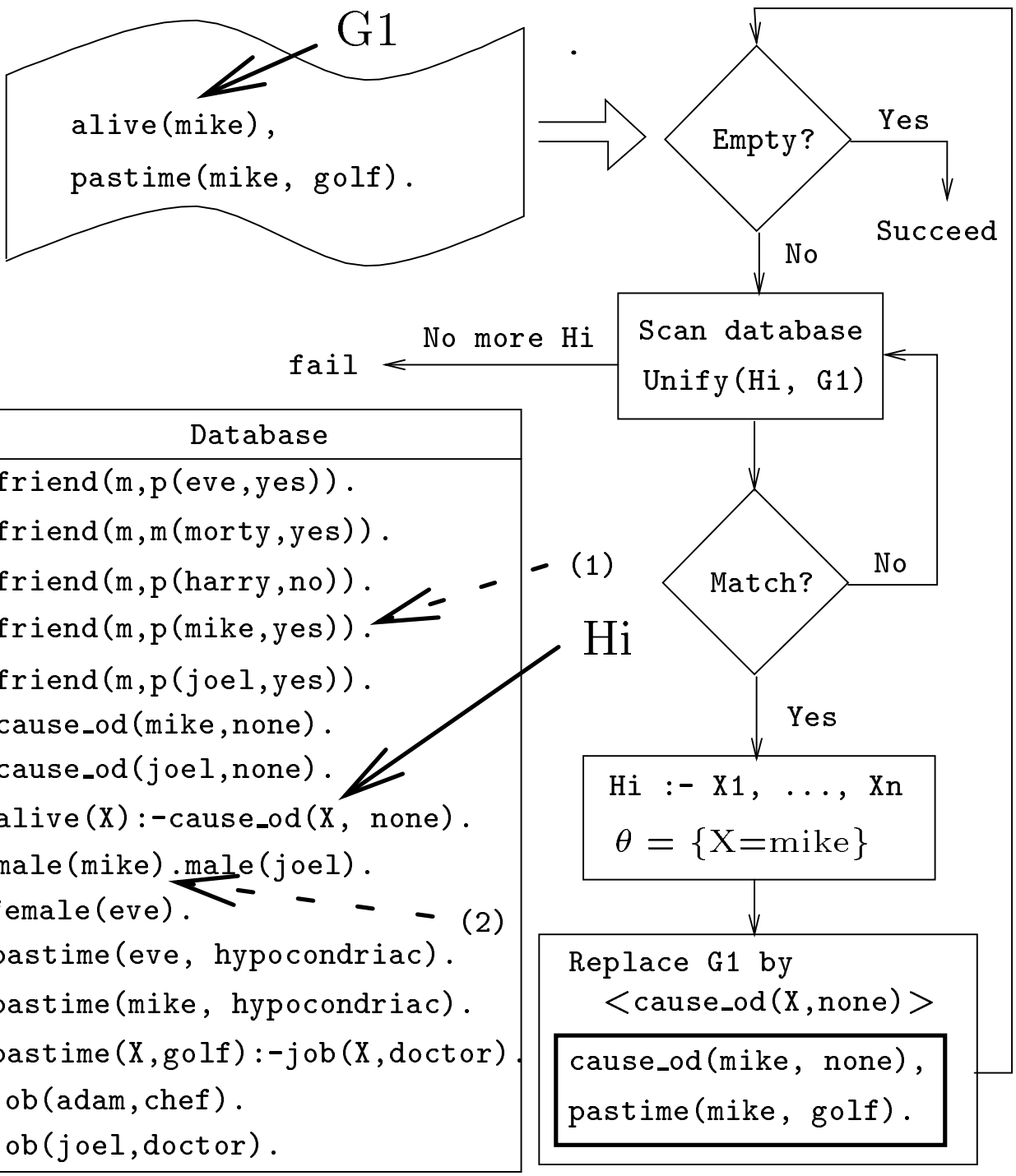
Execution Example V



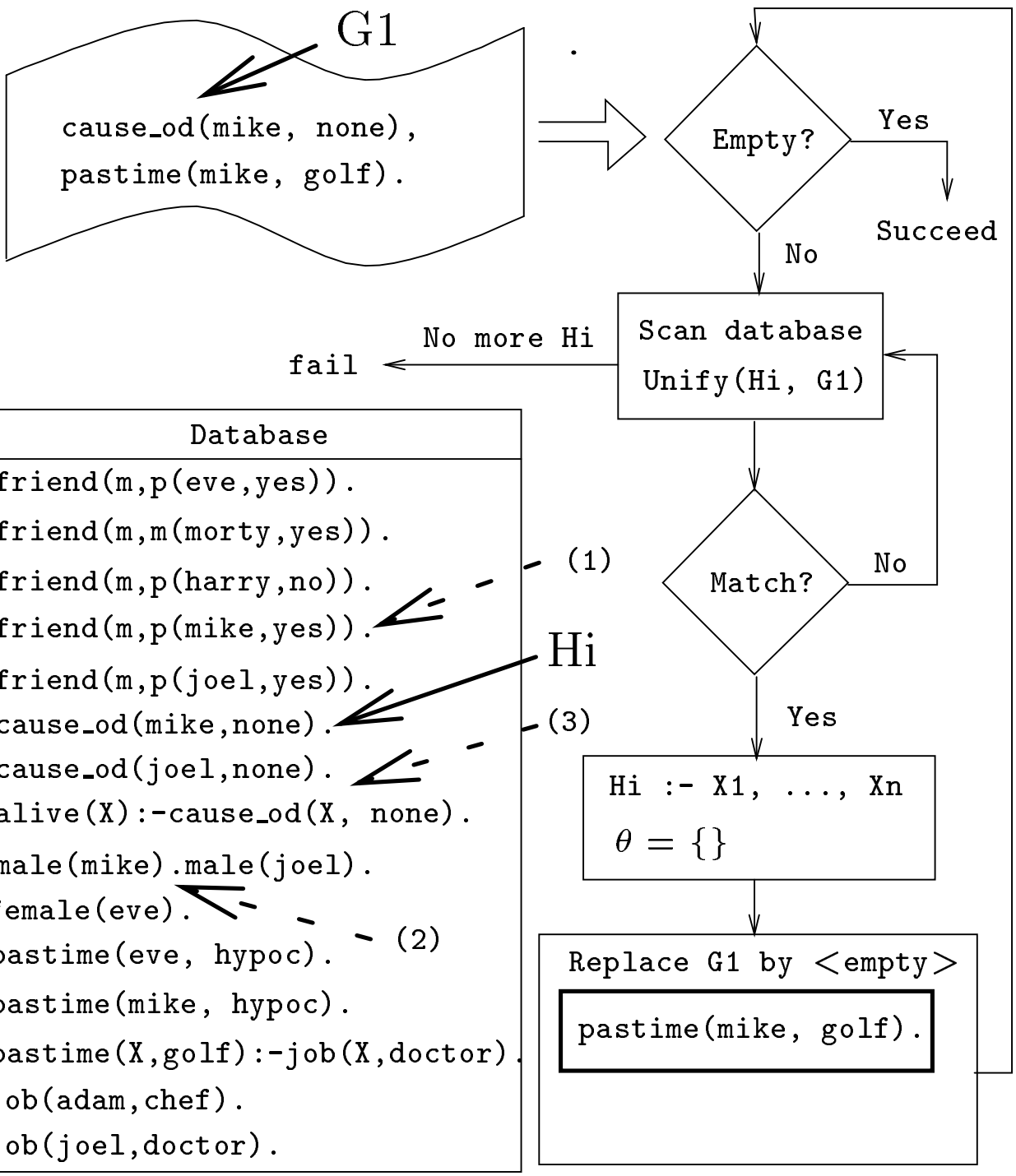
Execution Example VI



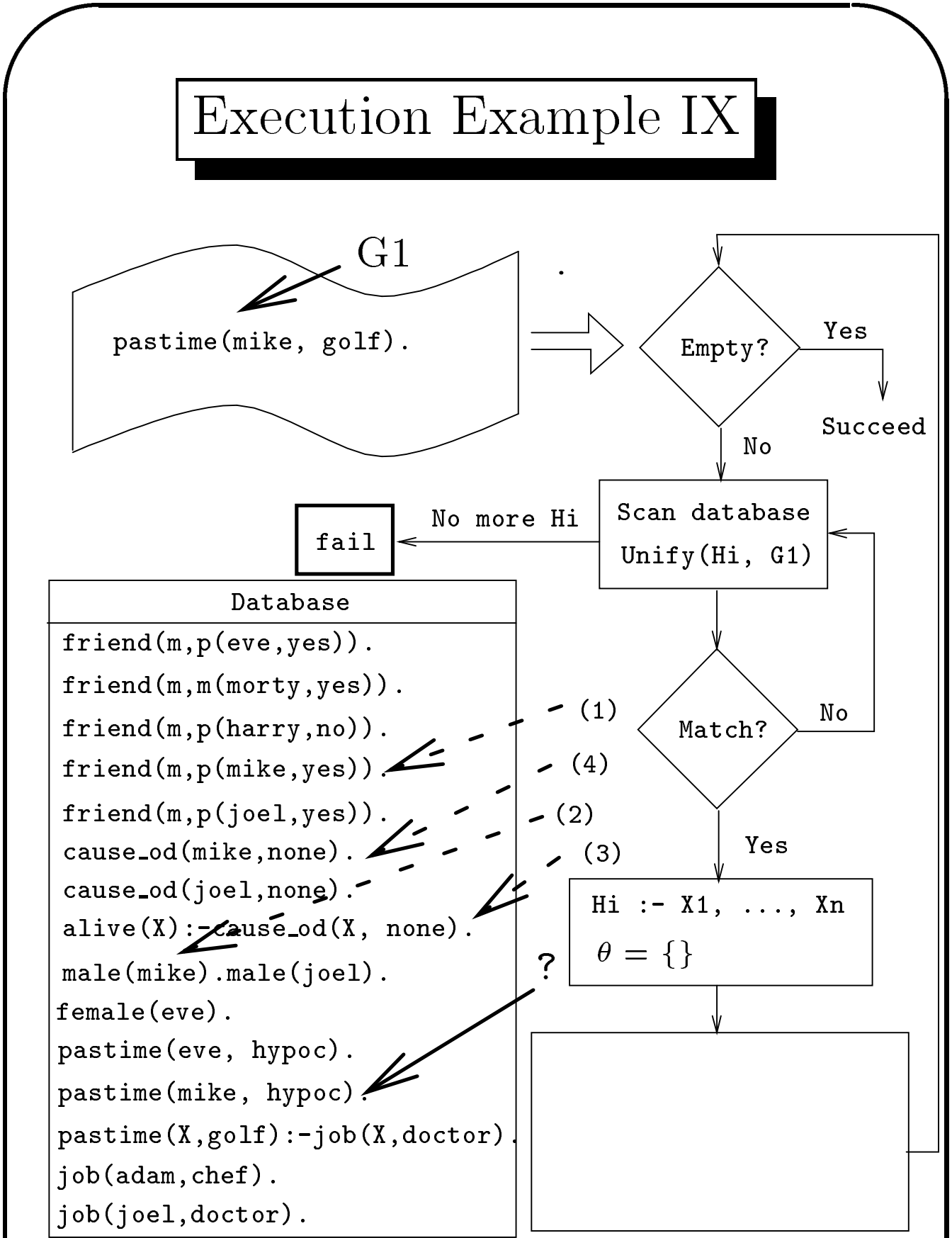
Execution Example VII



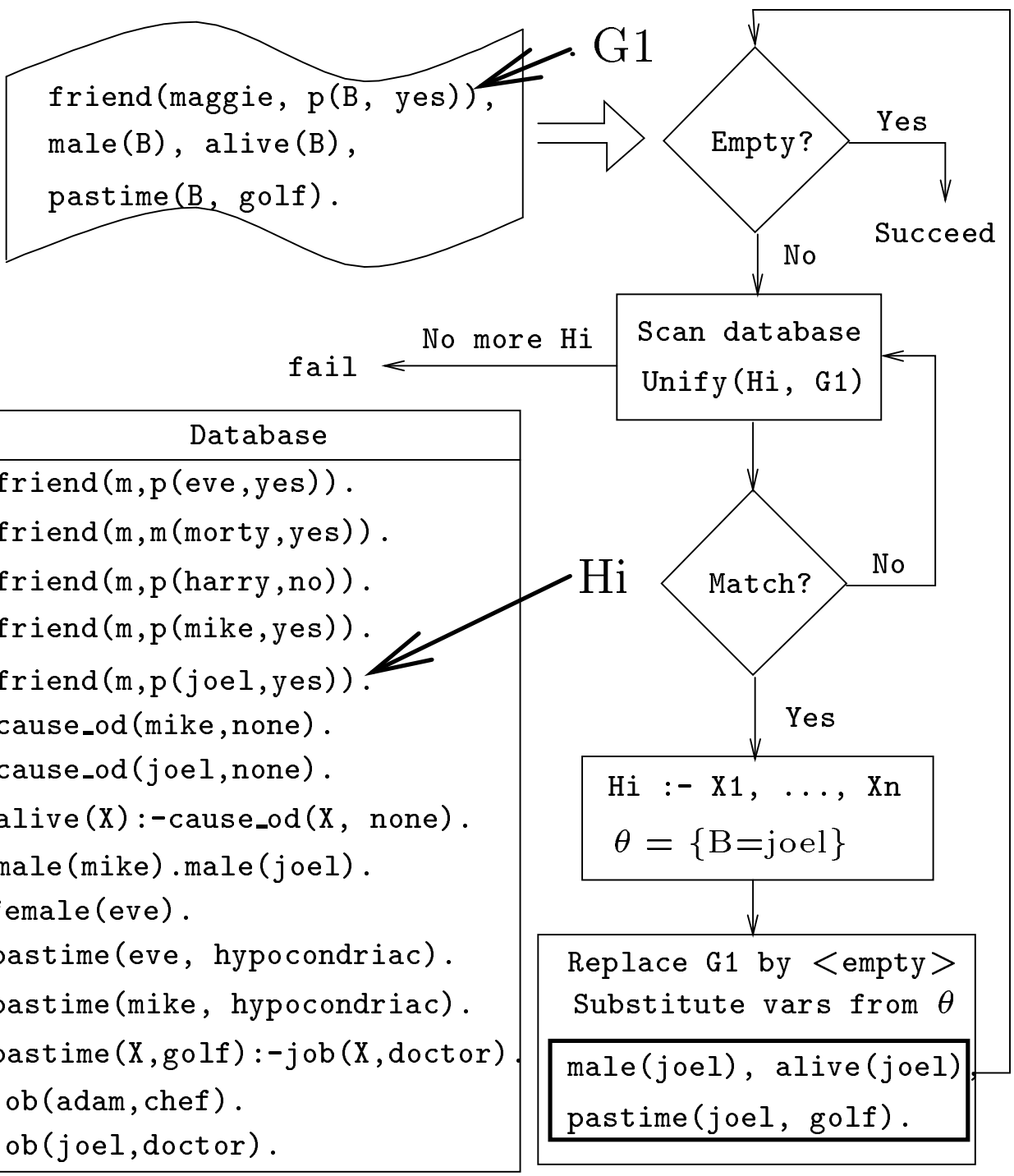
Execution Example VIII



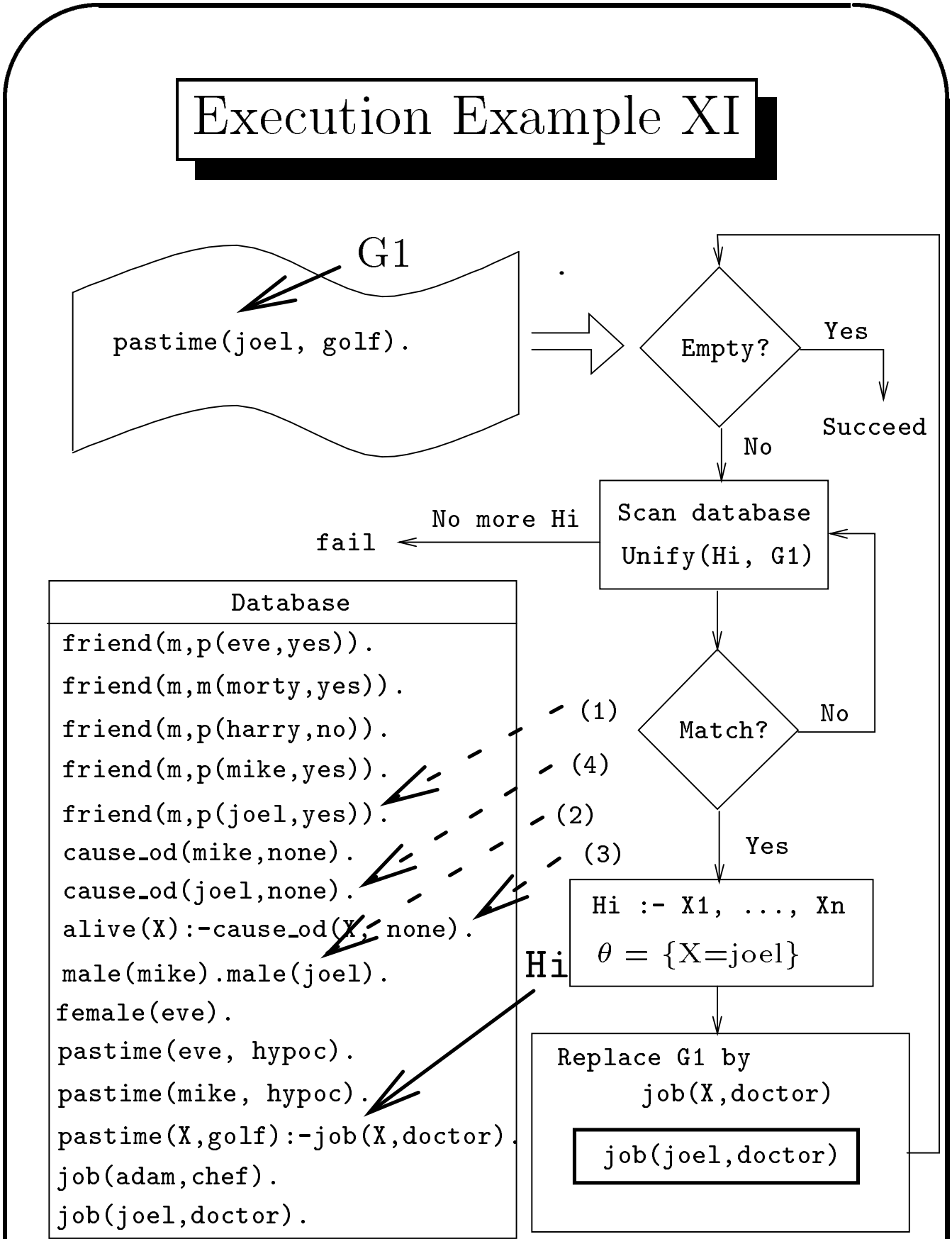
Execution Example IX



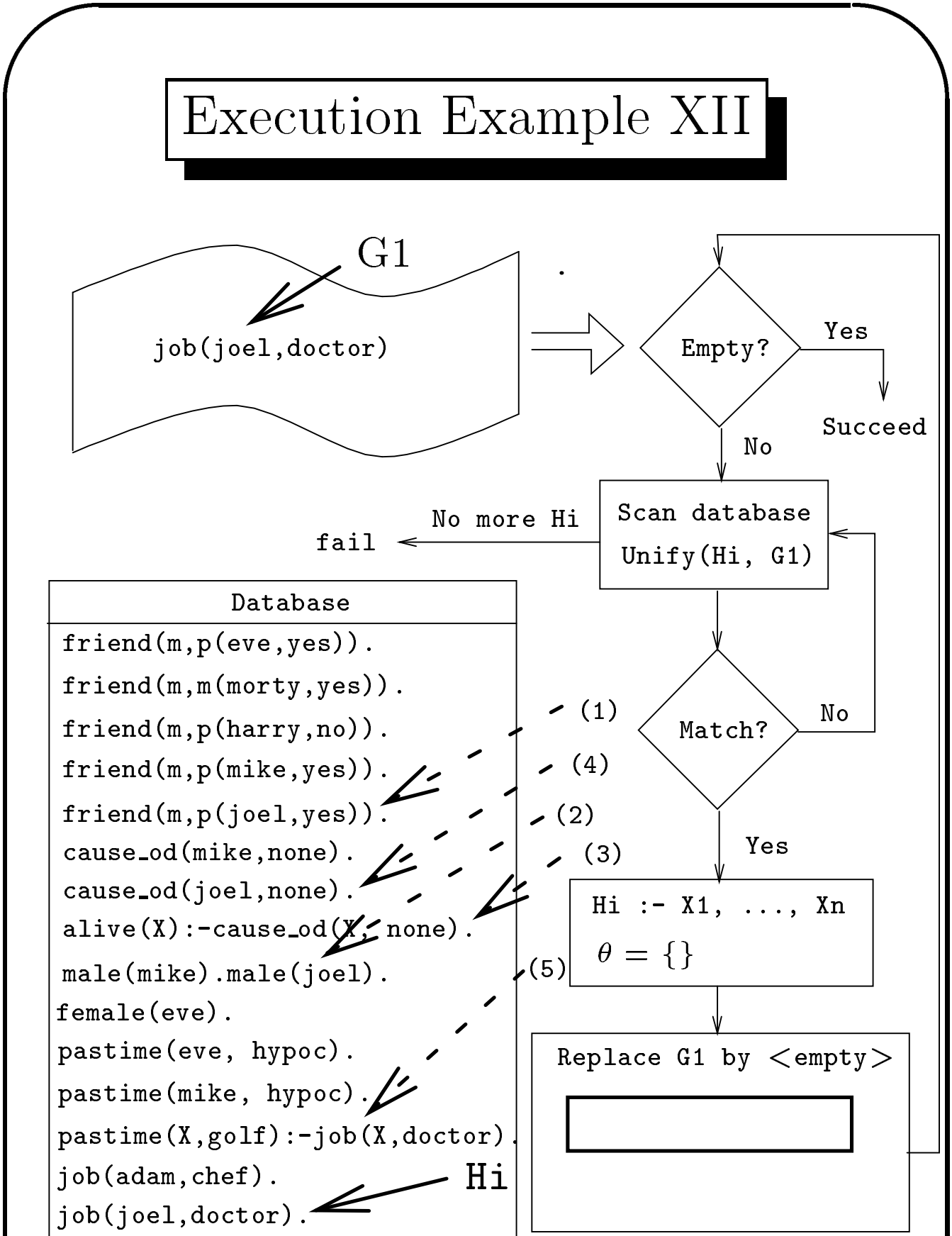
Execution Example X



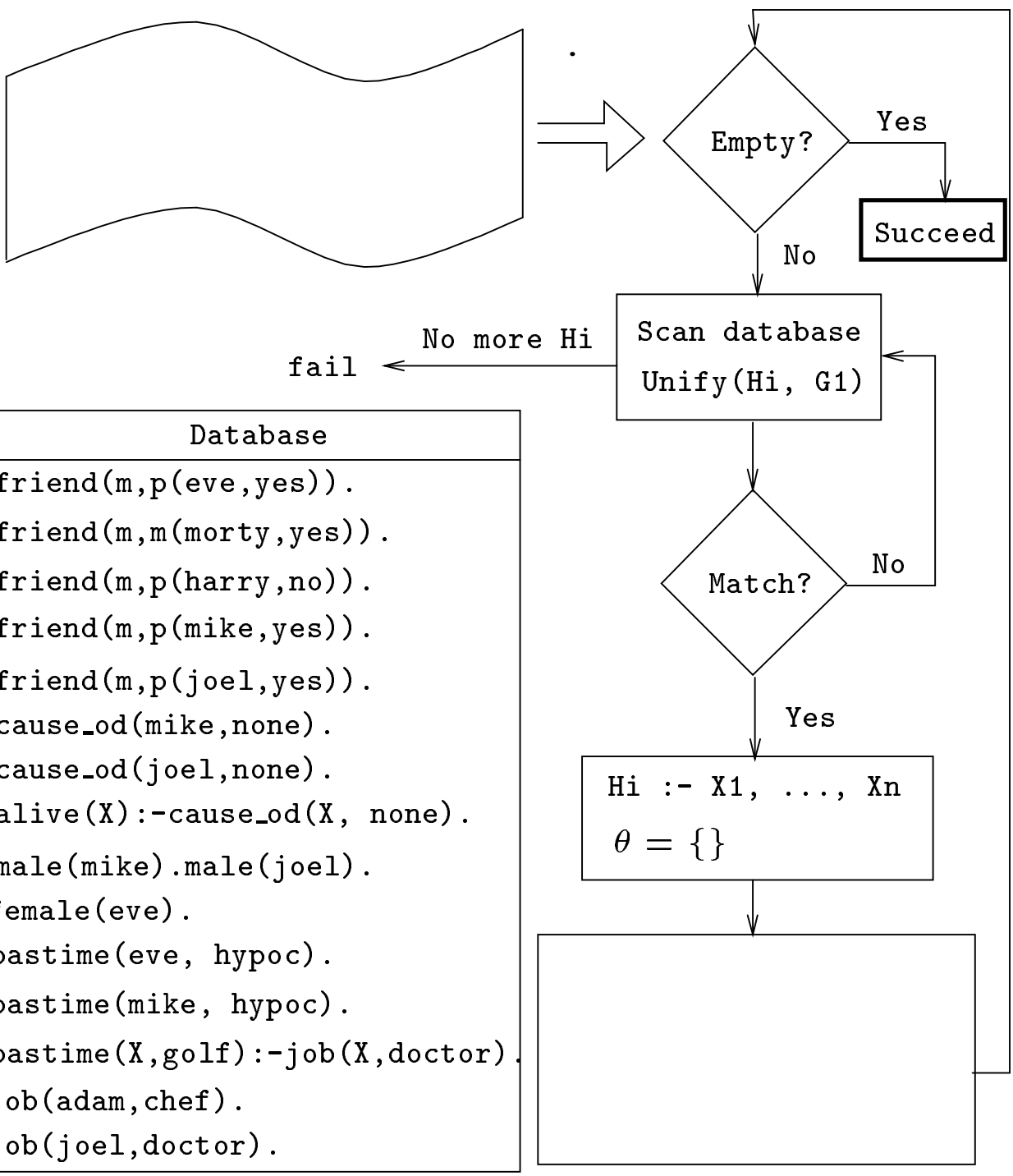
Execution Example XI



Execution Example XII



Execution Example XIII



Database
friend(m,p(eve,yes)).
friend(m,m(morty,yes)).
friend(m,p(harry,no)).
friend(m,p(mike,yes)).
friend(m,p(joel,yes)).
cause_od(mike,none).
cause_od(joel,none).
alive(X):-cause_od(X, none).
male(mike).male(joel).
female(eve).
pastime(eve, hypoc).
pastime(mike, hypoc).
pastime(X,golf):-job(X,doctor).
job(adam,chef).
job(joel,doctor).

Prolog So Far III

- A term is either a
  - a constant (an atom or integer)
  - a variable
  - a structure
- Two terms *match* if
  - there exists a variable substitution θ which makes the terms identical.
- Once a variable becomes instantiated, it stays instantiated.
- Backtracking *undoes* variable instantiations.
- Prolog searches the database sequentially (from top to bottom) until a matching clause is found.