CSc 372

Comparative Programming Languages

21 : Prolog — Execution

Department of Computer Science
University of Arizona

collberg@gmail.com

Christian Collberg

Execution

Executing Prolog

- Now that we know about matching, we can take a closer look at how Prolog tries to satisfy goals.
- In general, to solve a goal

$$G = G_1, G_2, \cdots, G_m,$$

  Prolog will first try to solve the sub-goal G_1.
- It solves a sub-goal G_1 it will look for a rule

$$H_i \text{ :- } B_1, \cdots, B_n$$

  in the database, such that G_1 and H_i will match.
- Any variable substitutions resulting from the match will be stored in a variable θ.

Executing Prolog...

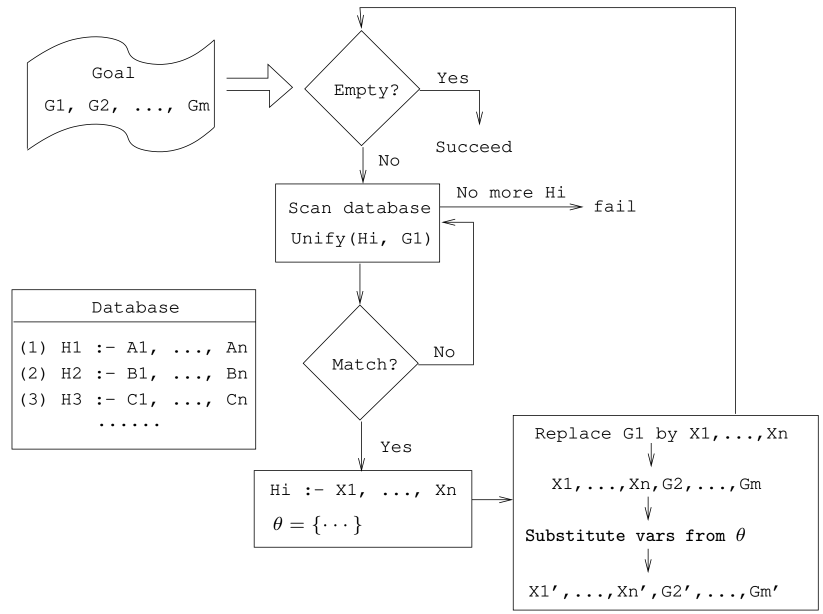
- A new goal will be constructed by replacing G_1 with $B_1, \cdots, B_n$, yielding

$$G' = B_1, \cdots, B_n, G_2, \cdots, G_m.$$

  If $n = 0$ the new goal will be shorter and we'll be one step closer to a solution to G!
- Any new variable bindings from θ are applied to the new goal, yielding G''.
- We recursively try to find a solution to G''.

Executing Prolog...

```
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, $\theta$) THEN
            $G'$ := $B_1, \cdots, B_n, G_2, \cdots, G_m$;
            $G''$ := substitute($G'$, $\theta$);
            Execute($G''$, Result);
         ENDIF;
         $i$ := $i + 1$;
      ENDDO
   ENDIF
```



Example

Northern Exposure Example

```
% 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)).
```

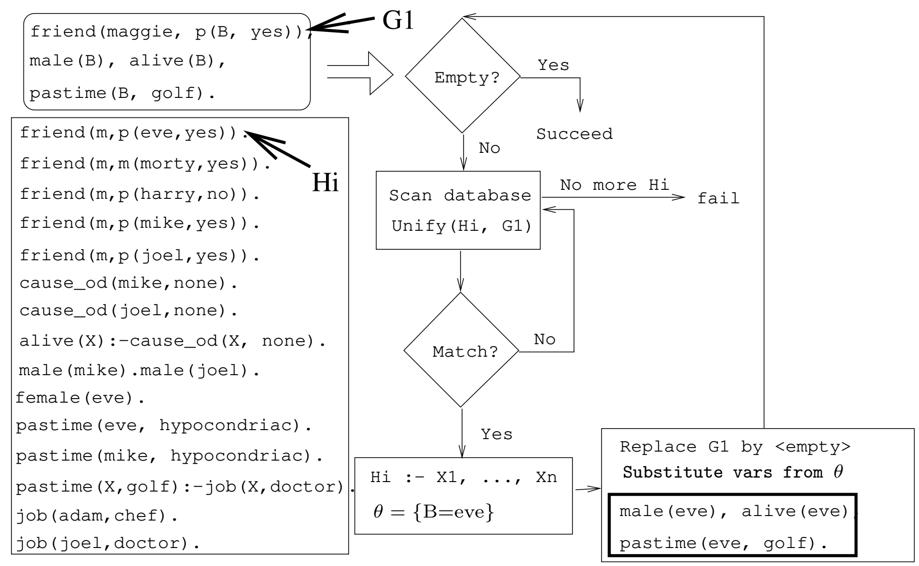
Maggie (Janine Turner)

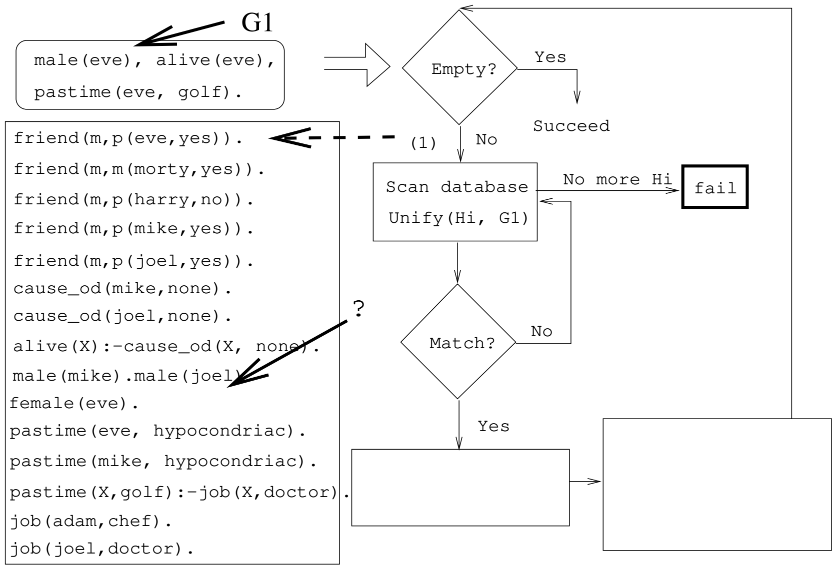
Northern Exposure Example. . .

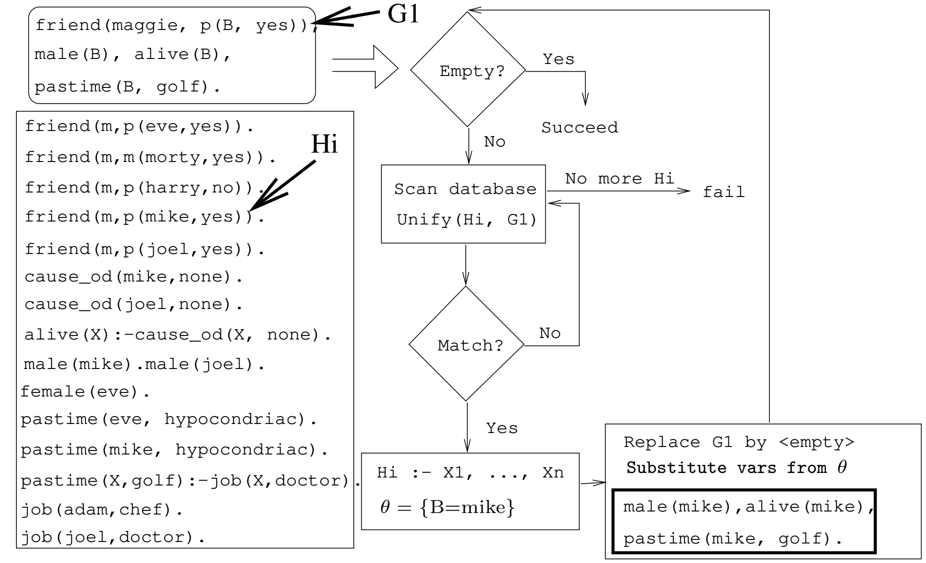
```
cause_of_death(morty, copper_deficiency).
cause_of_death(harry, potato_salad).
cause_of_death(bruce, fishing_accident).
cause_of_death(glenn, missile).
cause_of_death(dave, hypothermia).
cause_of_death(rick, hit_by_satellite).
cause_of_death(mike, none_yet).
cause_of_death(joel, none_yet).

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

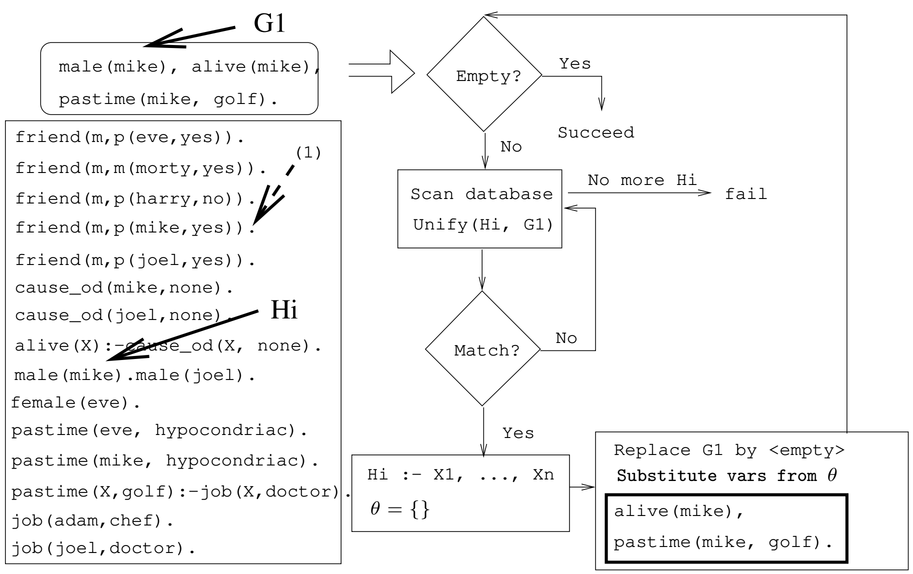
Northern Exposure Example. . .

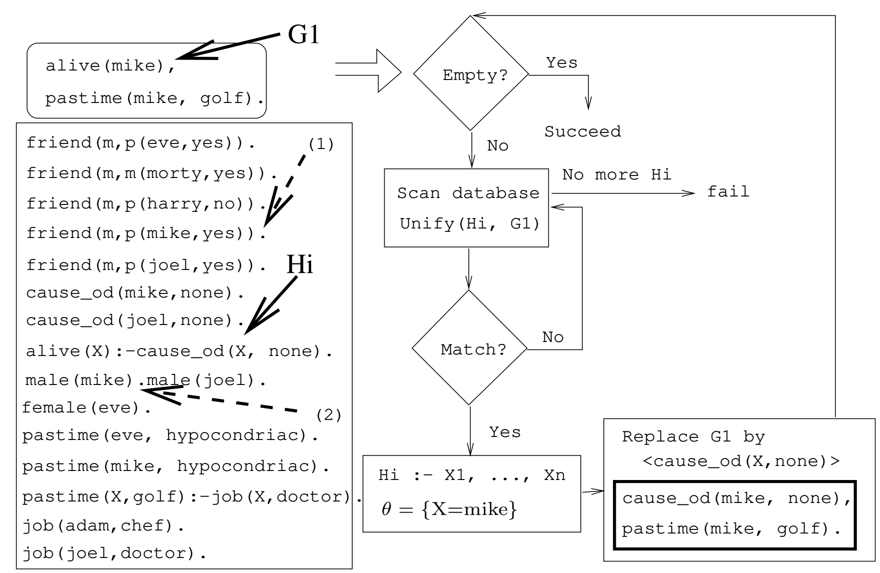
```
alive(X) :- cause_of_death(X, none_yet).

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).
```

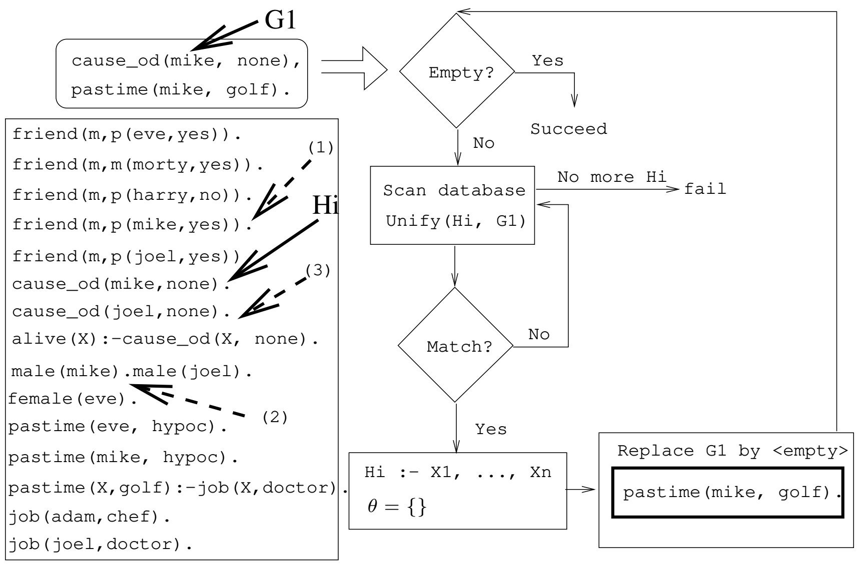


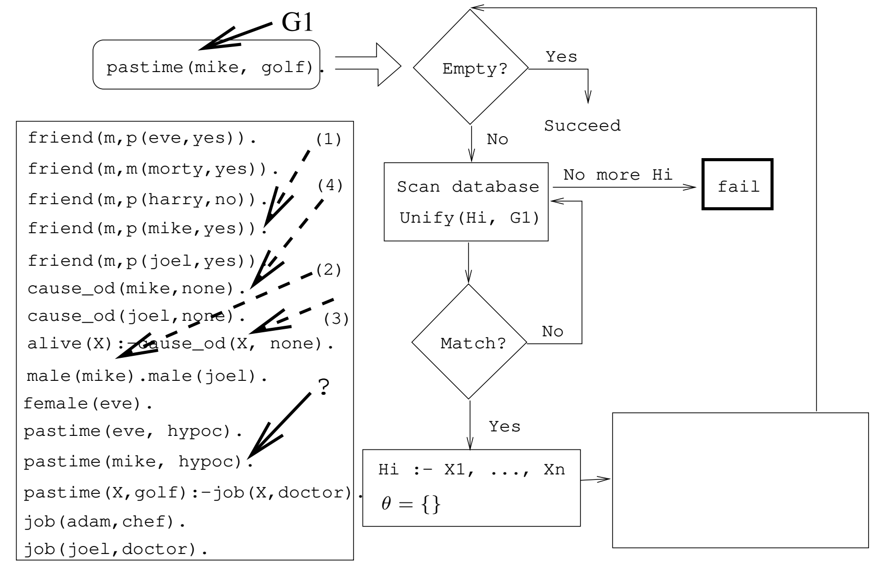

G1
male(eve), alive(eve),
pastime(eve, golf).
Empty?
Yes
Succeed
No
Scan database
Unify(Hi, G1)
No more Hi
fail
Match?
No
Yes
(1)
?
friend(m,p(eve,yes)).
friend(m,m(morty,yes)).
friend(m,p(harry,no)).
friend(m,p(mike,yes)).
friend(m,p(joel,yes)).
cause_od(mike,none).
cause_od(joel,none).
alive(X):-cause_od(X, none).
male(mike).male(joel).
female(eve).
pastime(eve, hypocondriac).
pastime(mike, hypocondriac).
pastime(X,golf):-job(X,doctor).
job(adam,chef).
job(joel,doctor).


G1
friend(maggie, p(B, yes)),
male(B), alive(B),
pastime(B, golf).
Empty?
Yes
Succeed
No
Scan database
Unify(Hi, G1)
No more Hi
fail
Match?
No
Yes
Hi
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, hypocondriac).
pastime(mike, hypocondriac).
pastime(X,golf):-job(X,doctor).
job(adam,chef).
job(joel,doctor).
Hi :- X1, ..., Xn
$\theta = \{B{=}mike\}$
Replace G1 by <empty>
Substitute vars from θ
male(mike),alive(mike),
pastime(mike, golf).


G1
male(mike), alive(mike),
pastime(mike, golf).
Empty?
Yes
Succeed
No
Scan database
Unify(Hi, G1)
No more Hi
fail
Match?
No
Yes
(1)
Hi
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, hypocondriac).
pastime(mike, hypocondriac).
pastime(X,golf):-job(X,doctor).
job(adam,chef).
job(joel,doctor).
Hi :- X1, ..., Xn
$\theta = \{\}$
Replace G1 by <empty>
Substitute vars from θ
alive(mike),
pastime(mike, golf).


G1
alive(mike),
pastime(mike, golf).
Empty?
Yes
Succeed
No
Scan database
Unify(Hi, G1)
No more Hi
fail
Match?
No
Yes
(1)
Hi
(2)
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, hypocondriac).
pastime(mike, hypocondriac).
pastime(X,golf):-job(X,doctor).
job(adam,chef).
job(joel,doctor).
Hi :- X1, ..., Xn
$\theta = \{X{=}mike\}$
Replace G1 by
<cause_od(X,none)>
cause_od(mike, none),
pastime(mike, golf).



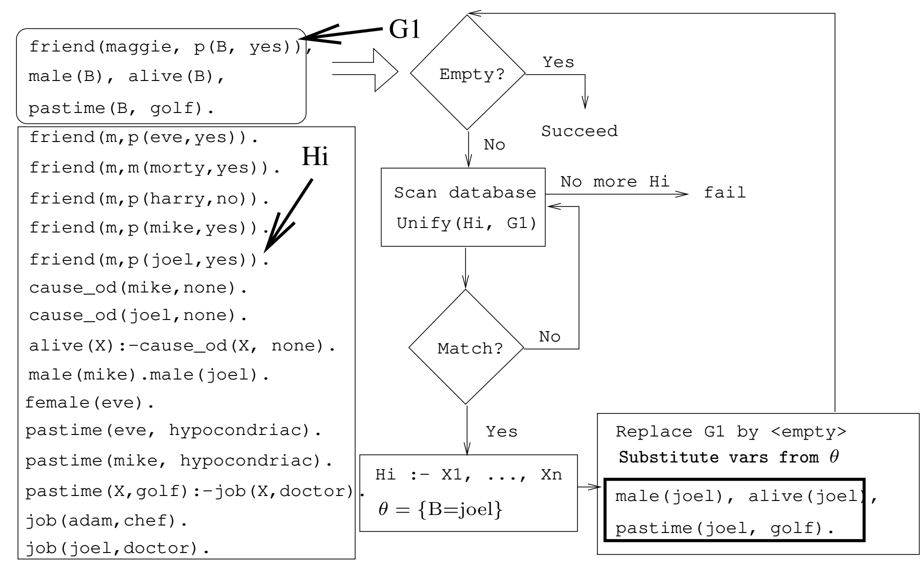


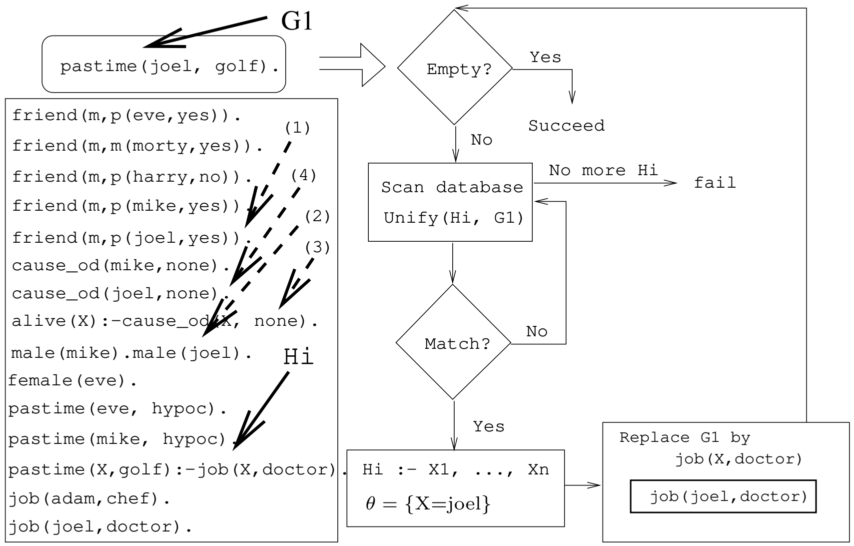
Northern Exposure Example. . .

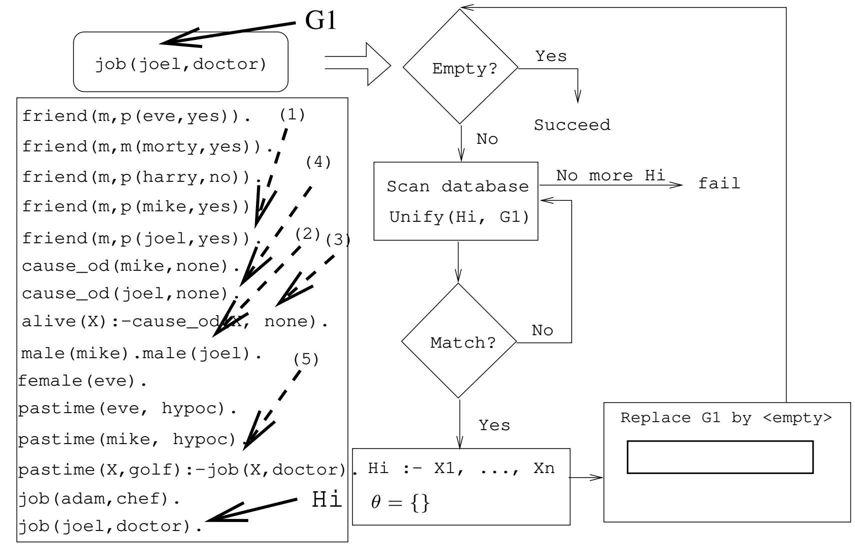
- We skip a step here.
- `pastime(mike, golf)` unifies with

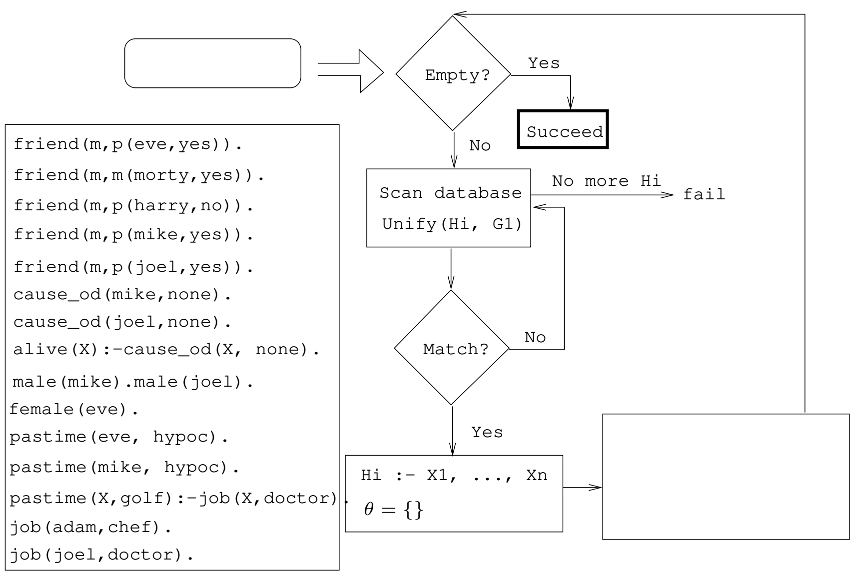
  `pastime(X, golf) :- job(X, doctor).`

  .
- However, `job(mike, doctor)` fails, and we backtrack all the way up to the original query.









Readings and References

- Read Clocksin-Mellish, Section 4.1.
- See http://www.moosefest.org for information about the annual Moosefest.
- See http://members.lycos.co.uk/janineturner/engl/index.html for pictures of Janine Turner, who plays Maggie.
- See http://home.comcast.net/~mcnotes/mcnotes.html for show transcripts.

Summary

Prolog So Far...

- 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.