

CSc 520

Principles of Programming Languages

44 : Logic Programming — Executing Prolog

Christian Collberg

collberg+520@gmail.com

Department of Computer Science
University of Arizona

Copyright © 2008 Christian Collberg

—Spring 2008 — 44

[1]

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

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

Prolog will first try to solve the sub-goal G_1 .

- It solves a sub-goal G_1 it will look for a rule

$$H_i :- B_1, \dots, B_n$$

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 .

520 —Spring 2008 — 44

[2]

Executing Prolog...

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

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

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'' .

—Spring 2008 — 44

[3]

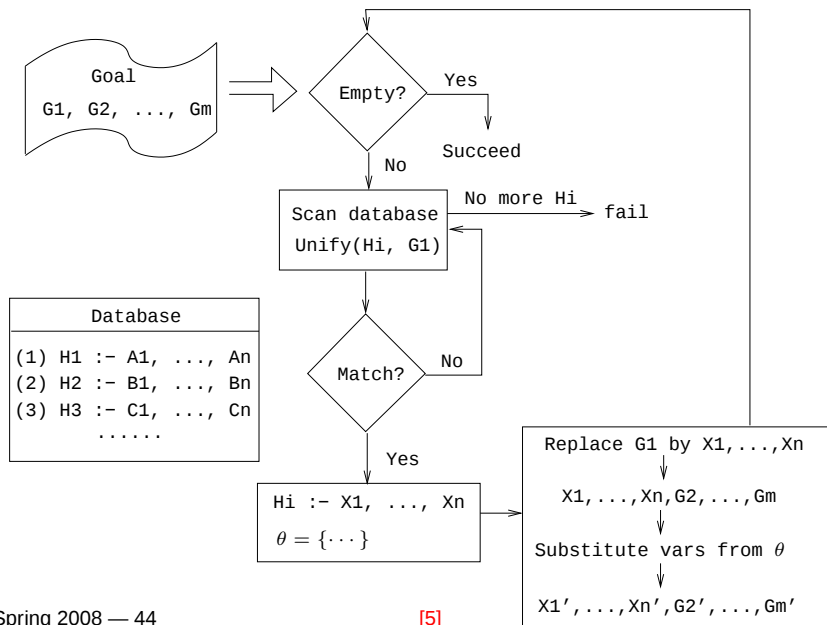
Executing Prolog...

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

520 —Spring 2008 — 44

[4]

Executing Prolog



—Spring 2008 — 44

[5]

Northern Exposure Example

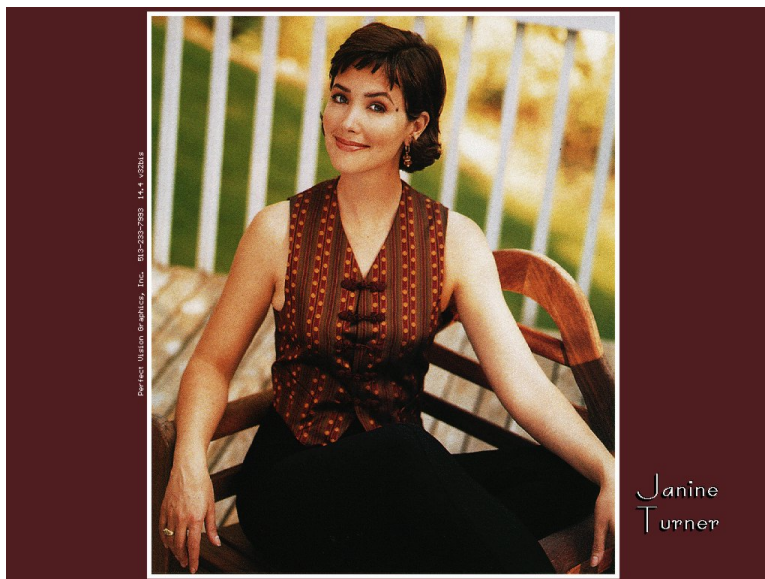
```

% 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)).
  
```

520 —Spring 2008 — 44

[6]

Maggie (Janine Turner)



—Spring 2008 — 44

[7]

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).
  
```

520 —Spring 2008 — 44

[8]

Northern Exposure Example...

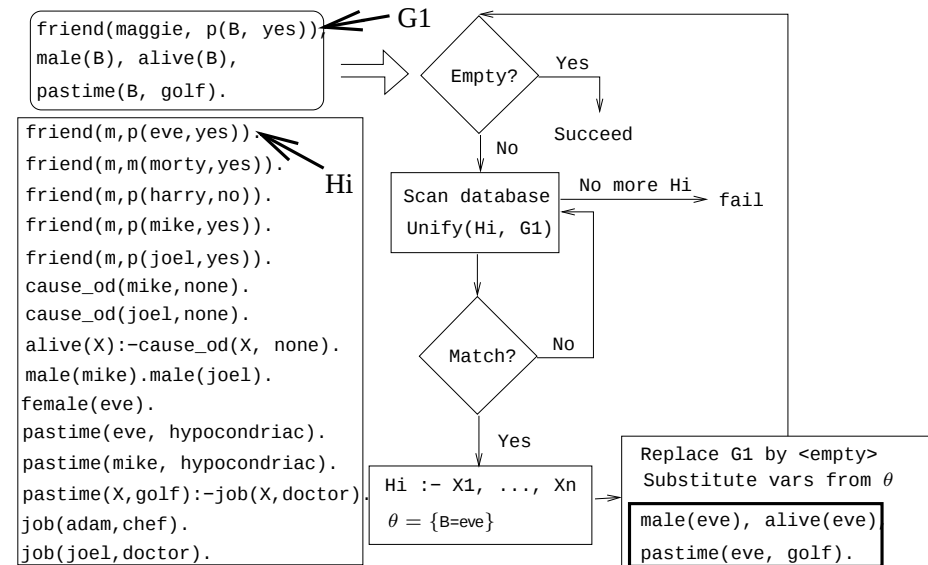
```
alive(X) :- cause_of_death(X, none_yet).
```

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

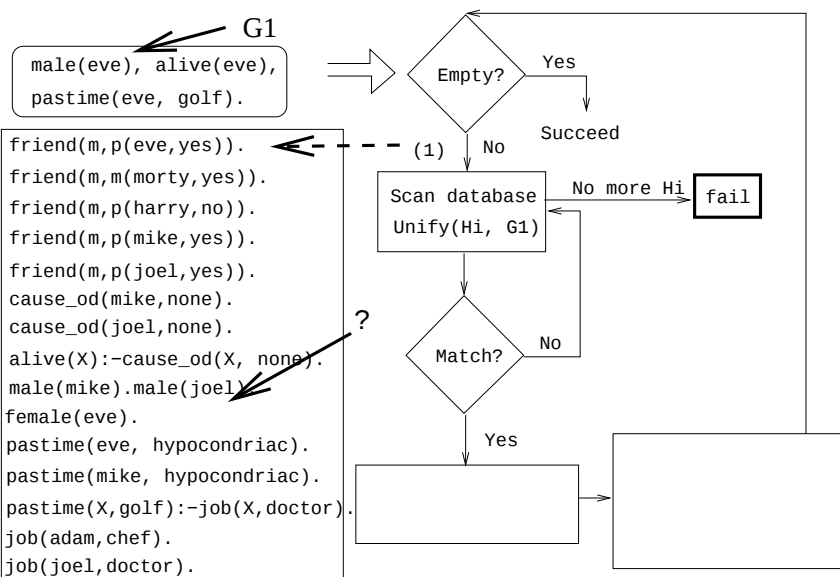
```
job(mike, lawyer).  job(adam, chef).
job(maggie, pilot). job(joel, doctor).
```

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

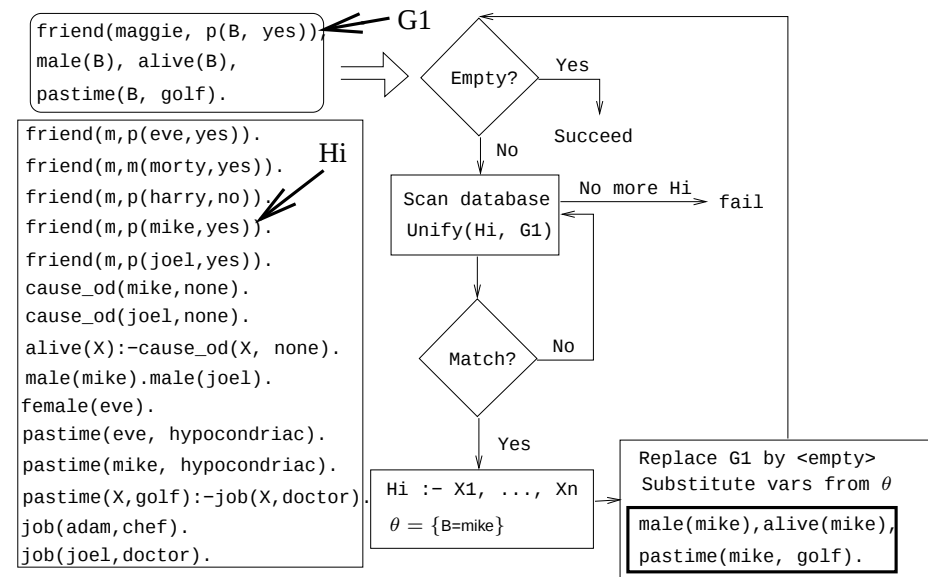
Northern Exposure Example...



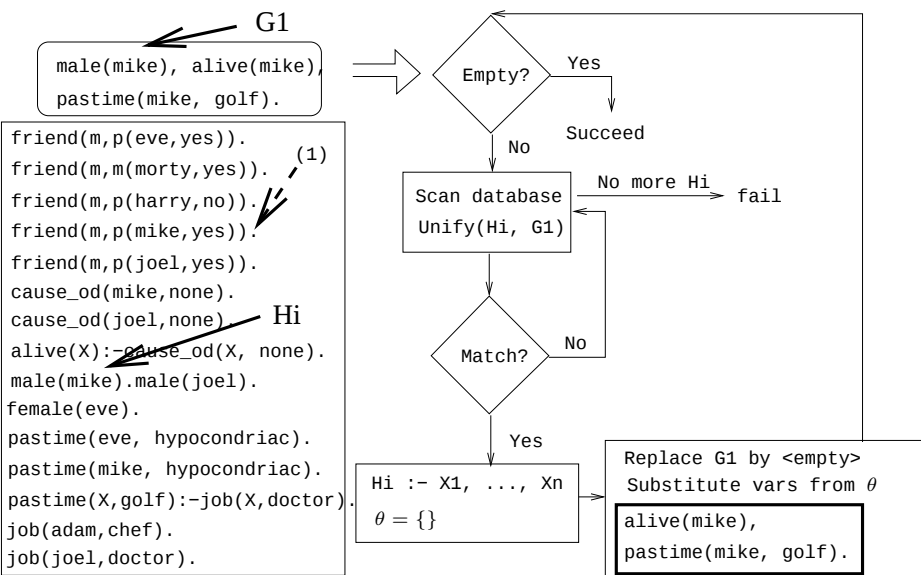
Northern Exposure Example...



Northern Exposure Example...



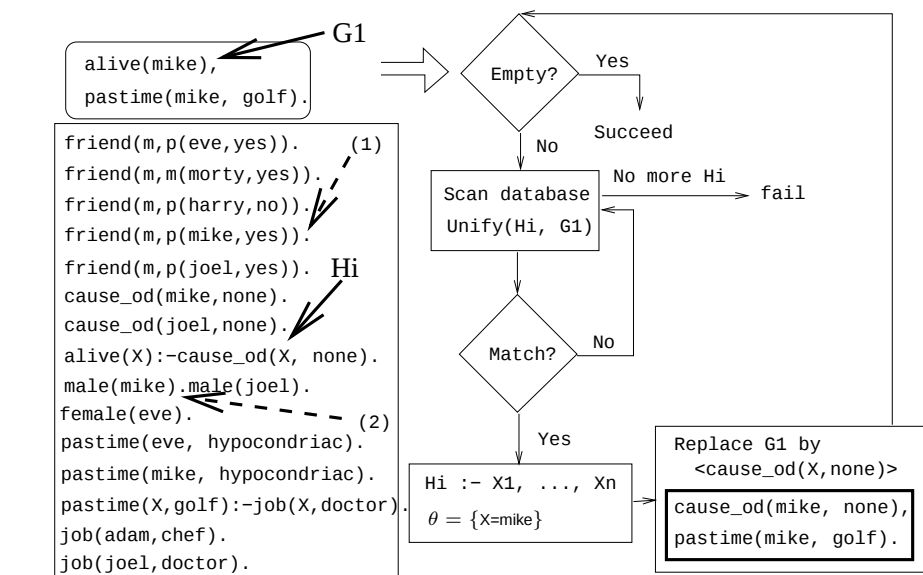
Northern Exposure Example...



—Spring 2008 — 44

[13]

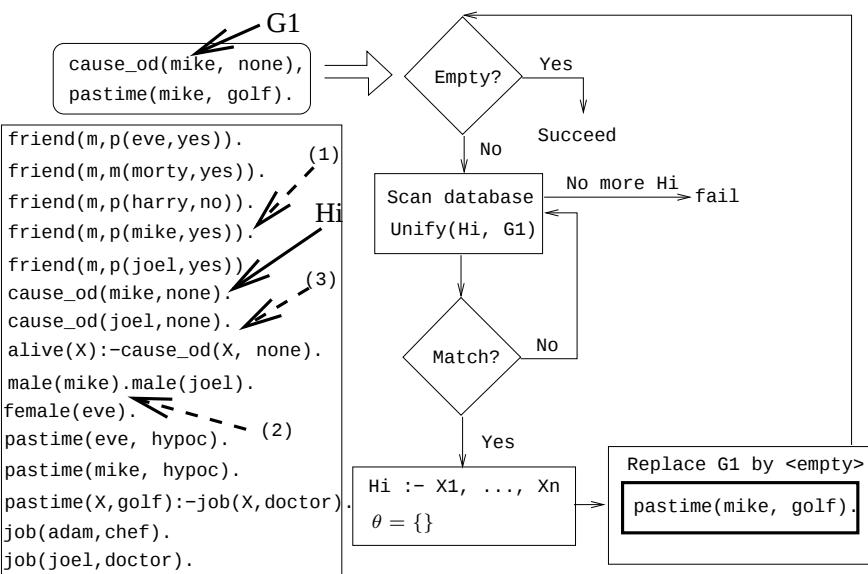
Northern Exposure Example...



520 —Spring 2008 — 44

[14]

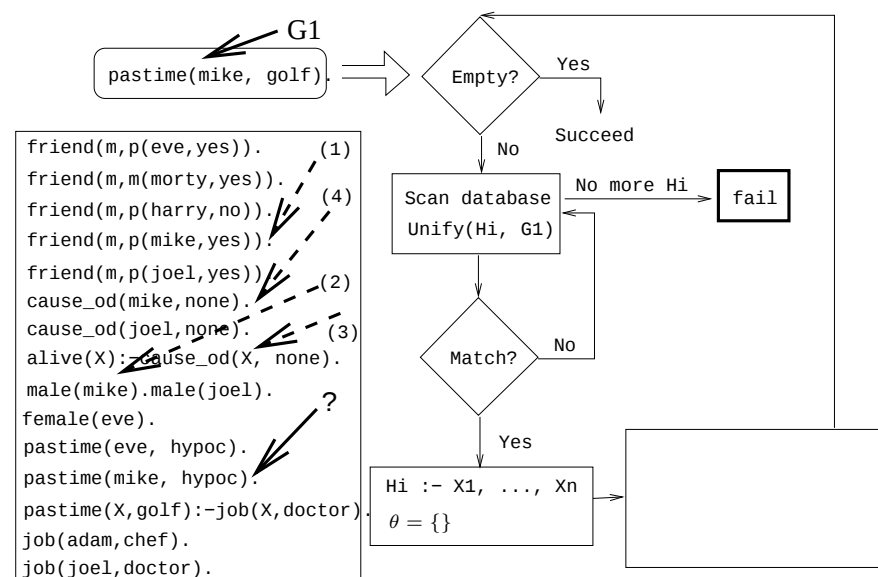
Northern Exposure Example...



—Spring 2008 — 44

[15]

Northern Exposure Example...



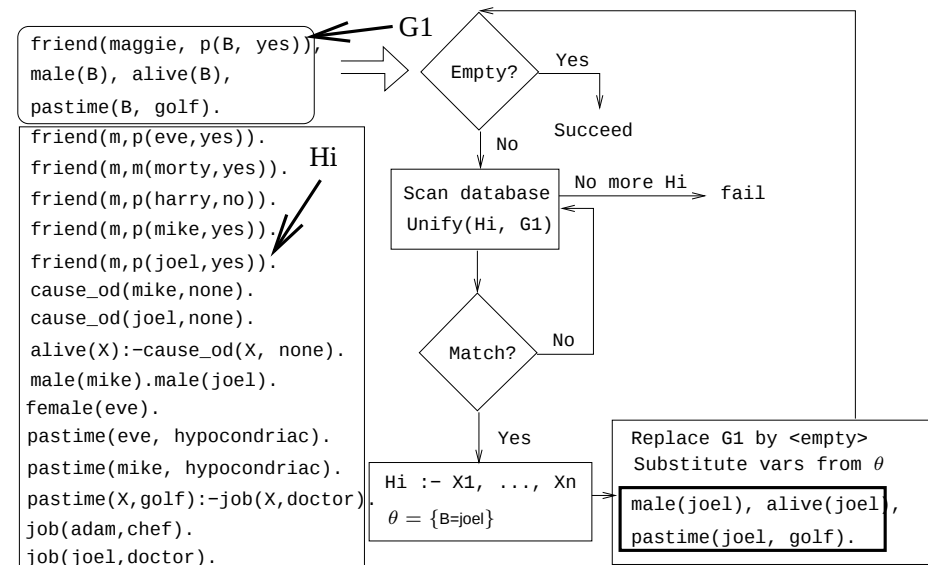
520 —Spring 2008 — 44

[16]

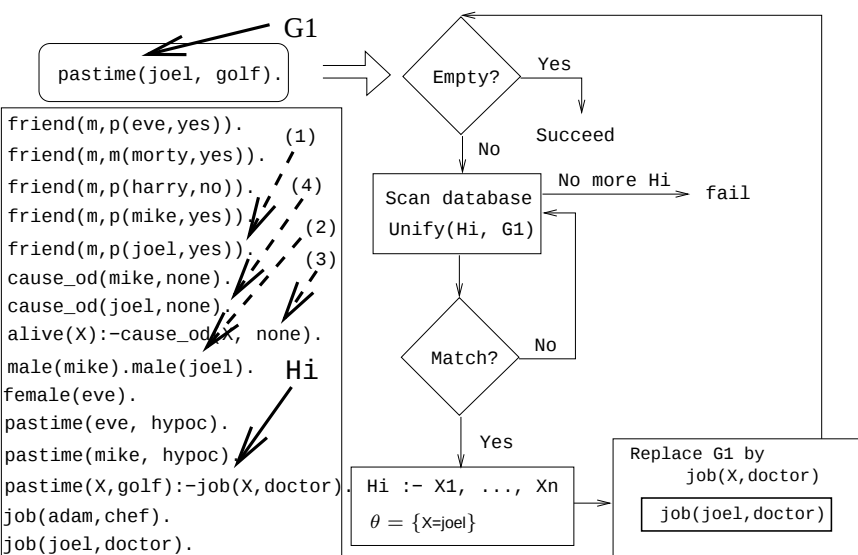
Northern Exposure Example...

- 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.

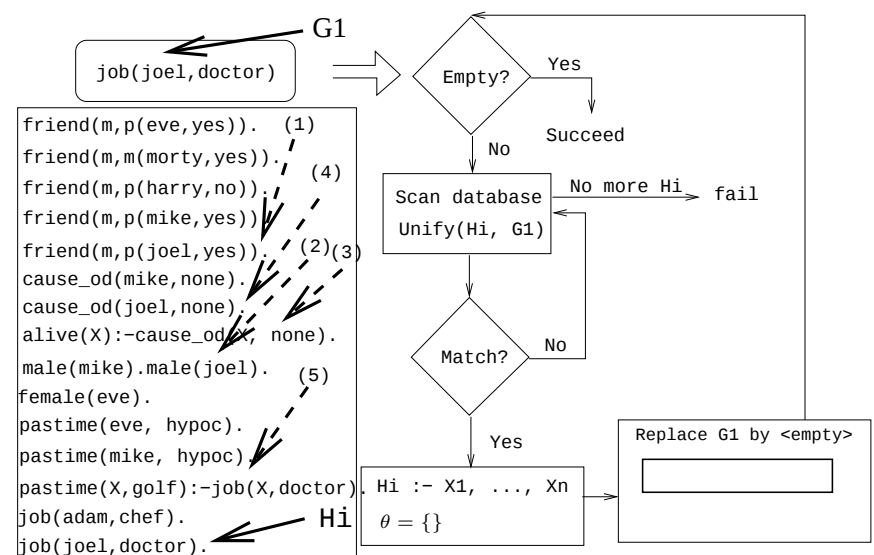
Northern Exposure Example...



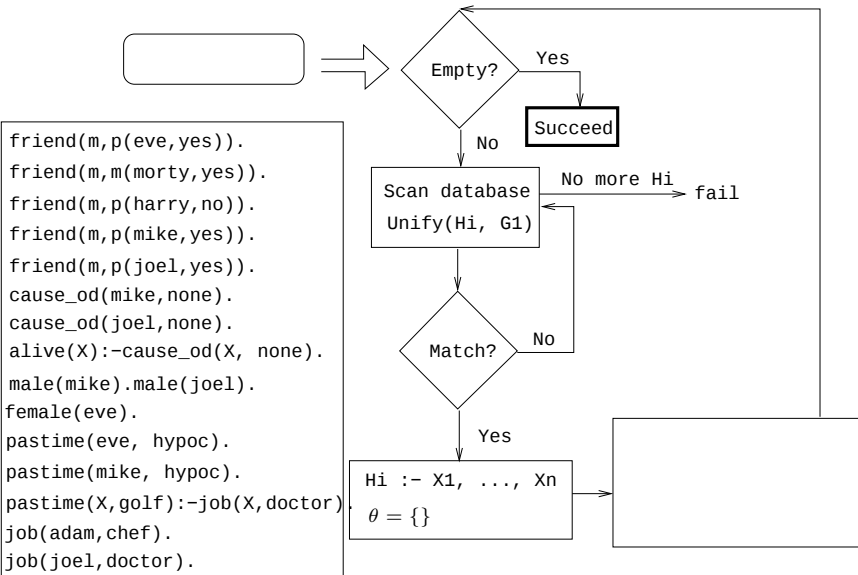
Northern Exposure Example...



Northern Exposure Example...



Northern Exposure Example...



—Spring 2008 — 44

[21]

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.

520 —Spring 2008 — 44

[22]

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.

—Spring 2008 — 44

[23]