Programming in Logic: Prolog

Meta-Interpreters

Readings: 23.2-3

Generating Proof Trees

- In pure Prolog, queries can be viewed as theorems to be proved & the KB viewed as a collection of axioms.
- With this perspective, when Prolog attempts to show that the current KB satisfies the query, it can be viewed as searching for a proof for that query.

Example: Giving Gifts

- **Axioms (aka Domain Theory):**

gives(P1, P2, G) :- likes(P1, P2), wouldPlease(G,P2).

gives(P1,P2,G) :- feelsSorryFor(P1,P2), wouldComfort(G,P2).

wouldPlease(G,P) :- needs(P,G).

wouldComfort(G,P) :- likes(P,G).

feelsSorryFor(P1,P2) :- likes(P1,P2), sad(P2).

feelsSorryFor(P,P) :- sad(P).

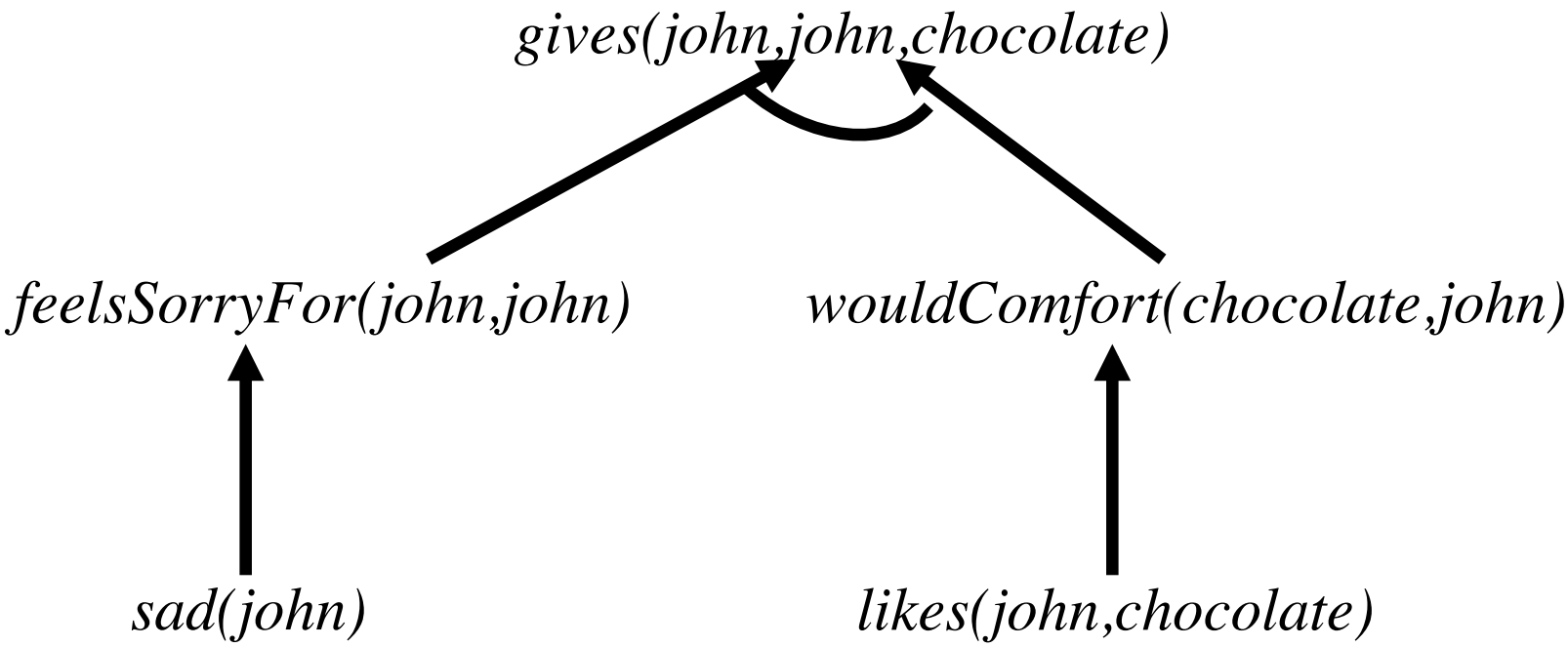
likes(john,annie). *likes(annie,john).*

likes(john,chocolate). *needs(annie,tennisRacket).*

sad(john).

- **Want to use axioms to prove:** *gives(john,john,chocolate)*

Proof for *gives(john,john,chocolate)*



Prolog Code for Capturing Proof

```
:- op(500, xfy, <==).

prove(true,true).

prove((Goal1,Goal2), (Proof1, Proof2)) :-
                prove(Goal1, Proof1),
                prove(Goal2 , Proof2).

prove(Goal, Goal <== Proof) :-
                clause(Goal,Body),
                prove(Body,Proof).
```

Transforming Proof into Rule

- From the proof, we see that whenever *sad(john)* and *likes(john,chocolate)* are facts in the KB then we can derive *gives(john,john,chocolate)*
- Can transform this into rule:

  gives(john,john,chocolate) :-
  sad(john), *likes(john,chocolate)*.
- Could also transform into:

  gives(john,john,chocolate) :-
  feelsSorryFor(john,john), *likes(john, chocolate)*.

Why transform it into rule?

- While the proof tree is short, it still could have taken Prolog a lot of search to find that proof.
- If we asserted that rule using *asserta/1*, then the next time we had the same situation we would find a proof directly.
- This is another example of rote learning.
- We can do better than this.

Generalizing the Query/Theorem

- Probably will have other queries about people giving themselves things (e.g., Microsoft employee buying himself a Porsche) that have the same sort of proof, e.g., they're sad and they bought themselves something they like.
- Generalize query from *gives(john,john,chocolate)* to *gives(Person,Person,Thing)*.
- Now redo that proof using these variables.

Generalized Rule

- The generalized proof allows us to create a more general rule:

  gives(Person, Person,Thing) :-

  sad(Person), *likes(Person, Thing)*.

- This rule can be used in a lot more situations than the original rule.
- Could generalize rule even more by moving up the proof tree to collect condition.

Operationality

- How do we indicate where in the proof tree to gather the body of the rule?
- The idea is that certain goals will be cheap to check while others will be expensive.
- The rule should avoid recomputing expensive goals.
- Goals that are cheap to check are *operational*.