Adding Disjunction to Datalog

Want to allow disjunction in body of clause

\[ a \leftarrow b \lor c \]

- Means that if either \( b \) or \( c \) is true, then \( a \) must be true

- Not increasing the power of datalog, just making it easier for knowledge engineers

First, how do we represent definite clauses in Python?

- As a list whose first element is head and rest are conjuncts of body

Definite Goals

Depth Bounded

Disjunction

Meta-Interpreters

How to find proofs in datalog

- Depth first search
- Breadth first search
- A* search
- Interactive Deepening
- Iterative Deepening

- Reasoning procedure
- Meta-language: what your programmer programs in to implement the reasoning procedure
- Base language: what your knowledge engineer programs in
- Textbook writes these different interpreters in Prolog
- Messy, as both base language and meta-language based on datalog
- Base language: what your knowledge engineer programs in
- Meta language: what your programmer programs in to implement the reasoning procedure
Example

- For each rule in KB, if the head is a disjunction, pick all disjuncts as new neighbors.
- Need to find all rules whose head matches a.

To find all neighbors of a:
  - Note that new disjunctions of variables which contain a
  - Disjunction replaces a with one of the disjuncts
  - Conjunction replaces a with entire conjunction
  - This is not the don't care non-determinism of picking which conjunct to work on first
  - Note that which disjunct we pick does matter

New representation might be a conjunction or disjunction:
  - Negated resolution is to replace a with a list of conjuncts q and if we
  - Negated resolution is to replace a with one of the disjuncts

Before (without disjunction):

Interpreter for Disjunction

- For simplicity, let's not allow embedded bodies. Just:
  - a rule with 'and' or 'or' as head followed by bodies is a body
  - atoms are bodies

Examples:

```
[ [ p q a ] ]
[ [ q a ] ]
```

Body:

- Clause is a list of length 2, first element is head, second is body

Defining Clauses in Tcl

- Need a more complex representation to handle disjunction
Overview

- Disjunction
- Depth Bounded
- Delaying Goals

Depth Bounded Reasoning Procedure

- Similar to Iterative Deepening
- But you don’t keep going to deeper and deeper depths
- Is it alwaysSound and complete?
- Will it always halt?
Implemention

- Rather than always choosing the first atom in the conjunction:
  - Have rules for when you can skip over atoms
  - At each step of the goal, keep rechecking whether atoms at the front of the answer clause (which were previously delayed) can be proved
  - Do not move delayed atoms to end of answer clauses, as you should respect the defined ordering as much as possible
  - If there are any ground atoms, you might want to prove those first
  - While we are at it...
  - If there are any ground atoms, you might want to prove those first
  - Do not move delayed atoms to end of answer clauses, as you should

Example

- Delaying might allow a definite clause to be used in a way it wasn't anticipated for
- Example: Brother, where we expect both variables to be inputs:

```
brother(X,Y) :-
  not(X = Y),
  sonof(X,Z),
  sonof(Y,Z),
```

- When used where either is not inputs, you get wrong behavior:
```
brother(X,Y) :-
  not(X = Y),
  sonof(X,Z),
  sonof(Y,Z),
  X = Y,
```

Delaying Goals

- Some goals, rather than being proved, can be delayed
  - To delay assumptions, so that you can collect assumptions that are needed
  - To delay goals that are or(X,Y) in which X is not fully instantiated
Recap of Class

• Adding Disjunction
  - Did not change expressiveness of Datalog

• Depth Bounded Reasoning
  - Reasoning procedure that always halts in a certain amount of time
  - Sacrifices completeness

• Delaying Goals
  - Mechanism to remove reliance from knowledge engineer to order
  - Can speed up the reasoning procedure

Caveat

• Re-ordering of atoms works in Datalog

• Re-ordering of atoms does not make sense in Prolog in general
  - Re-ordering of atoms does not make sense in Prolog in general

• Re-ordering of atoms does not make sense in Prolog in general

• Prolog has assert/retract for adding/removing of facts during a proof

• Programmer explicitly controls ordering of conjuncts in rules and rules in KB and can use this ordering to take advantage of side effects from assert/retract

• Re-ordering of atoms does not make sense in Prolog in general