Meta-Interpreters

- How to find proofs in datalog
  - Depth first search
  - Breath first search
  - A* search
  - Interactive Deepening
- Textbook writes these different interpreters in Prolog
  - Messy, as both base language and metalanguage based on datalog
  - Base language: what your knowledge engineer programs in
  - Meta language: what your programmer programs in to implement the reasoning procedure
- We use Python for implementing reasoning procedure

Overview

⇒ Disjunction
• Depth Bounded
• Delaying Goals
Defining Clauses in Python

• Need a more complex representation to handle disjunction
  - Clause is a list of length 2, first element is head, second is body

• Body:
  - atoms are bodies
  - a list with ‘and’ or ‘or’ as head followed by bodies is a body

• Examples:
  
  \[
  s \leftarrow a \wedge b \ \\
  s \leftarrow a \vee b \\
  s \leftarrow a \wedge (b \vee c \vee (e \wedge f) \vee g) \wedge h
  \]

• For simplicity, lets not allow embedded bodies. Just:
  
  ['s', ['and', 'a', 'b']]
Interpreter for Disjunction

• Before (without disjunction)
  - State of a proof was a conjunction
  - Used resolution to replace conjunct $a$ with a list of conjuncts $b \lor c \lor d$, if we had the rule $a \leftarrow b \land c \land d$

• Now, replacement might be a conjunction or disjunction
  - Keep answer clause to be a conjunction of atoms
  - Conjunction: replace $a$ with entire conjunction
  - Disjunction: replace $a$ with one of the disjuncts
  - Note that which disjunct we pick does matter
  + This is not don’t care non-determinism of picking which conjunct to work on first

• To find all of the neighbors of $a$
  - Need to find all rules whose head matches $a$
  - For each rule that is a disjunction, add answer clause for each disjuncts

Example

$KB$:

$\text{yes } \leftarrow \text{[and a k l m]}$

• What are the new neighbors?
Depth Bounded Reasoning Procedure

• Similar to Iterative Deepening
  But you don’t keep going to deeper and deeper depths

• Could be done for either depth-first or breadth-first

• Will it always halt?

• Is it sound and complete?
Overview

• Disjunction
• Depth Bounded
⇒ Delaying Goals

Delaying Goals

• Some goals, rather than being proved, can be delayed
  - Delay subgoals with variables, in the hope that subsequent calls will ground the variables
    + Delay an is(X, Y) in which Y is not fully instantiated
  - Delay assumptions, so that you can collect assumptions that are needed to prove a goal
    + We will see more of this later in the course
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

  \[
  \text{brother}(X,Y) :- \\
  \text{not}(X = Y), \\
  \text{sonof}(X,Z), \\
  \text{sonof}(Y,Z),
  \]

- When used where either is not inputs, you get wrong behavior
  - Have reasoning procedure delay evaluation of ‘not =’ until both parameters are instantiated

Implementation

- Rather than always choosing the first atom in the conjunction
  - Have rules for when you can skip over atoms
  - At each step of the proof, 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 clause, as you should respect the defined ordering as much as possible

- While we are at it ...
  - If there are any ground atoms, you might want to prove those first
    - As no variable bindings from earlier atoms will not affect their truth
  - If they can’t be proved, may as well find out sooner than later
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 conjuncts in a rule
  - 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
  - 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