Complete Knowledge Assumption

- Why not assume KB includes all positive facts, and everything else is false (similar to unique name assumption)
  
- Examples:
  - If I haven't stated that two rooms are adjacent, assume they are not
  - If I haven't stated that Jim is Mary's father, assume he is not

- We don't want to state negative facts in KB, but we want to ask queries of form \( \neg p \)
  and use \( \neg p \) in body of rule

- How can we formalize complete knowledge assumption?

Horne Clauses

- Allowed negative information to be expressed
- Some things are false
- Some things are true
- Some things I just don't know
- System can conclude now that

Overview

Approach 1: Clark's Completion

Approach 2: Negation as Failure

Complete Knowledge Assumption
Variable Case: Example

- Example
  - student(mary)
  - student(john)
  - student(ying)

- Same as
  - student(X) ← X = mary
  - student(X) ← X = john
  - student(X) ← X = ying

- Note that `=' sign could mean equality

- Collect them all together and you get
  - student(X) ← X = mary ∨ X = john ∨ X = ying

- Completion is
  - student(X) ↔ X = mary ∨ X = john ∨ X = ying

Clark's Completion

- Typically only used with Datalog

- If you have
  - a → b
  - a → c
  - a → d

- You have equivalently
  - a → (b ∨ c ∨ d)

- Clark's Normal Form
  - Clark's Completion: a ↔ (b ∨ c ∨ d)

- If you have predicate p defined by no clauses in KB
  - The completion is p → false
  - Which is the same as saying ¬p

Overview

- Complete Knowledge Assumption
  - Approach 1: Clark's Completion
  - Approach 2: Negation as Failure
Using Clark's Completion

• Typically just used with Datalog

Clark Completion though requires:
- Disjunctive and negative knowledge
- Usually also assume UNA or need axioms for equality

• Can be applied to just some of the predicates, not necessarily all
- Only use rules from KB that have predicate on left hand side
- Do not use ones in which it is on the right hand side

Several Clauses

• Say if you have in Clark Normal form

p(V_1,\ldots,V_n) \leftarrow C_1 \ldots p(V_1,\ldots,V_n) \leftarrow C_n

• Clark completion of \( p \) is
\[ p(V_1,\ldots,V_n) \leftrightarrow C_1 \lor \ldots \lor C_n \]

- Note that each \( C_i \) might have a number of conjunctions to it for its variable bindings
- Before putting it together, make sure each part doesn't have any other variables in common other than the \( V_i \)'s

Variable Case

• Example
\[ p(t_1,\ldots,t_n) \leftarrow B \]

• Clark Normal form is
\[ p(V_1,\ldots,V_n) \leftarrow V_1 = t_1 \land \ldots \land V_n = t_n \land B \]

• Clark's Completion is
\[ p(V_1,\ldots,V_n) \leftrightarrow V_1 = t_1 \land \ldots \land V_n = t_n \land B \]

Say if you have in Clark Normal form

Example
Overview

- Approach 1: Clark's Completion
- Approach 2: Negation as Failure

Complete Knowledge Assumption

Example

Can be used with recursive predicates

Unique Name Assumption

Can I prove ¬ls(s(0),0)

New Axioms

lt(0,s(X))

lt(s(X),s(Y)) ← ls(X,Y)

Clark Normal Form:

lt(A,B) ← A=0 ∧ B=s(X)

lt(A,B) ← A=s(X) ∧ B=s(Y) ∧ ls(X,Y)

Clark Completion:

ls(A,B) ↔ (A=0 ∧ B=s(X)) ∨ (A=s(X) ∧ B=s(Y) ∧ ls(X,Y))

New rule to add to KB:

lt(A,B) ← ls(A,B)

Convert to CNF:

lt(A,B) ← (A=0 ∧ B=s(X)) ∨ (A=s(X) ∧ B=s(Y) ∧ ls(X,Y)) ∨ ¬ls(A,B)

New Axioms

A=0 ∨ A=s(X) ∨ ¬ls(A,B) A=0 ∨ B=s(Y) ∨ ¬ls(A,B)

A=0 ∨ ls(X,Y) ∨ ¬ls(A,B) B=s(X) ∨ A=s(X) ∨ ¬ls(A,B)

B=s(X) ∨ B=s(Y) ∨ ¬ls(A,B) B=s(X) ∨ ls(X,Y) ∨ ¬ls(A,B)
Variables and Delaying

• Must be careful about variables - Similar to inequality, need to delay till variables are instantiated
  - Prolog doesn't do this
• Example:
  \[
  \text{KB} = \{ p(X) \leftarrow \neg q(X) \land r(X), q(a), q(b), r(b), r(d) \} = KB
  \]

Modifying a Top-Down Reasoning Procedure

• When see \( \neg p \), do recursive proof
  - If you can prove \( p \) then fail, otherwise succeed
• Example:
  \[
  \text{KB} = \{ p(X) \leftarrow r(X) \land \neg q(X), q(a), q(b), r(b), r(d) \} = KB
  \]

Negation as Failure

• A simpler way to make the CKA is to use negation as failure
  - Allow \( \neg q \) clauses
  - Allow \( \neg q \) to be used in bodies of clause clauses
  - Add negation as failure to database
  - \( \neg q \) can be used to assume \( p \) is false
  - A simpler way to make the CKA is to use negation as failure
  - If you cannot prove \( p \), assume \( \neg p \)