Overview

⇒ Complete Knowledge Assumption
  • Approach 1: Clark’s Completion
  • Approach 2: Negation as Failure

Horne Clauses

• Allowed negative information to be expressed
• System can conclude now that
  - some things are true
  - some things are false, using the negative information
  - some things it just doesn’t know
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 that they are not
  - If I haven’t stated that Jim is Mary’s father, assume that 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?

Overview

• Complete Knowledge Assumption
  \( \Rightarrow \) Approach 1: Clark’s Completion

• Approach 2: Negation as Failure
Clark’s Completion

- Typically only used with Datalog
- If you have
  \[ a \leftarrow b_1 \]
  ...
  \[ a \leftarrow b_n \]
- You have equivalently \[ a \leftarrow b_1 \lor ... \lor b_n \]
  - Clark’s Normal Form
- Clark’s Completion: \[ a \leftrightarrow (b_1 \lor ... \lor b_n) \]
- If you have predicate \( p \) defined by no clauses in KB
  - the completion is \( p \rightarrow \text{false} \)
  - Which is the same as saying \( \neg p \)

Variable Case: Example

- Example
  \[ \text{student}(\text{mary}) \]
  \[ \text{student}(\text{john}) \]
  \[ \text{student}(\text{ying}) \]
- Same as
  \[ \text{student}(X) \leftarrow X = \text{mary} \]
  \[ \text{student}(X) \leftarrow X = \text{john} \]
  \[ \text{student}(X) \leftarrow X = \text{ying} \]
  - Note that ‘\( = \)’ sign could means equality
- Collect them all together and you get
  \[ \text{student}(X) \leftarrow X = \text{mary} \lor X = \text{john} \lor X = \text{ying} \]
- Completion is
  \[ \text{student}(X) \leftrightarrow X = \text{mary} \lor X = \text{john} \lor X = \text{ying} \]
### 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
  \[ p(V_1, \ldots, V_n) \leftrightarrow V_1 = t_1 \land \ldots \land V_n = t_n \land B \]

### 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
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

Example 1

parent(X,Y) ← father(X,Y)
parent(X,Y) ← mother(X,Y)
parent(joe,hunter)

Let’s clark completion on parent
Example 2

- Can be used with recursive predicates

\[ \text{lt}(0, s(X)) \]
\[ \text{lt}(s(X), s(Y)) \leftarrow \text{ls}(X, Y) \]

**Clark Normal Form:**
\[ \text{lt}(A, B) \leftarrow A = 0 \land B = s(X) \]
\[ \text{lt}(A, B) \leftarrow A = s(X) \land B = s(Y) \land \text{ls}(X, Y) \]
\[ \text{lt}(A, B) \leftarrow (A = 0 \land B = s(X)) \lor (A = s(X) \land B = s(Y) \land \text{ls}(X, Y)) \]

**Clark Completion:**
\[ \text{ls}(A, B) \rightarrow (A = 0 \land B = s(X)) \lor (A = s(X) \land B = s(Y) \land \text{ls}(X, Y)) \]

**New rule to add to KB:**
\[ (A = 0 \land B = s(X)) \lor (A = s(X) \land B = s(Y) \land \text{ls}(X, Y)) \leftarrow \text{ls}(A, B) \]

**Convert to CNF:**
\[ (A = 0 \land B = s(X)) \lor (A = s(X) \land B = s(Y) \land \text{ls}(X, Y)) \lor \neg \text{ls}(A, B) \]

**New Axioms**
\[ A = 0 \lor A = s(X) \lor \neg \text{ls}(A, B) \]
\[ A = 0 \lor \text{ls}(X, Y) \lor \neg \text{ls}(A, B) \]
\[ B = s(X) \lor \text{ls}(X, Y) \lor \neg \text{ls}(A, B) \]

Can I prove \( \neg \text{ls}(s(0), 0) \)

yes \lor \text{ls}(s(0), 0)

use \( B = s(X) \lor B = s(Y) \lor \neg \text{ls}(A, B) \)

yes \lor 0 = s(X) \lor 0 = s(Y)

Unique Name Assumption
Overview

- Complete Knowledge Assumption
- Approach 1: Clark’s Completion
  ⇒ Approach 2: Negation as Failure

Negation as Failure

- A simpler way to make the CKA is to use negation as failure
  - If you cannot prove \( p \), assume it is false
  - Call this \( \sim p \)
- Add negation as failure to Datalog
  - Allow \( \sim p \) to be used in bodies of datalog clauses
  - Allow negative queries: \(? \sim p\)
Variables and Delaying

• Must be careful about variables
  - Similar to inequality, need to delay till variables are instantiated
  - Prolog doesn’t do this

• Example:
  $KB = \{p(X) \leftarrow q(X) \land \neg r(X), \quad q(a), \quad q(b), \quad r(b), \quad r(d)\}$
  $\n p(X) \n$