First Order Logic

Introduction to AI
Bert Huang

Review

- Propositional logic syntax and semantics
- Inference in propositional logic: table, inference rules, resolution
- Horn clauses, forward/backward chaining

First Order Logic

- Terminology: predicates, arity, terms, functions, atoms
- Quantifiers: universal, existential
- Inference: differences from propositional logic

Terminology

- Predicates: functions that map variables to truth values
  - is_woman(Marge), is_mother_of(Lisa, Marge)
  - “is_woman” is a **predicate**, “Marge” is a **term/constant**, and “is_woman(Marge)” is a **formula**
  - Can have any **arity**: binary, 3-ary, 4-ary…n-ary
- Functions map from terms to terms: e.g., motherOf(Bart)→Marge

Simple Propositionalization

- Define propositional logic variables for every atomic formula
  - M1 = is_mother_of(Bart, Homer), M2 = is_mother_of(Lisa, Snowball), …

    W1 = is_woman(Marge), W2 = is_woman(Lisa), W3 = is_woman(Homer), …
- Perform inference as in propositional logic

Child	Mother	is_mother_of
Bart	Homer	F
Lisa	Snowball	F
Bart	Marge	T
Homer	Mona	T

Name	is_woman
Lisa	T
Marge	T
Homer	F
Bart	F

Quantifiers

- More general knowledge: e.g., “all mothers are women”
  - ∀ x, ∀y, is_mother_of(x, y) → is_woman(y)
  - For all x and y, if y is x’s mother, then y is a woman.
  - Universal quantifier
- Existential quantifier: ∀x, ∃y, is_mother_of(x, y)
  - Every x has a mother

Propositionalizing is Expensive

Lisa
Maggie
Marge
Bart
Homer

is_mother_of

Lisa
Maggie
Marge
Bart
Homer

is_mother_of(Lisa, Lisa)
is_mother_of(Lisa, Maggie)
is_mother_of(Lisa, Marge)
is_mother_of(Lisa, Bart)
is_mother_of(Lisa, Homer)
is_mother_of(Maggie, Lisa)
is_mother_of(Maggie, Maggie)
is_mother_of(Maggie, Marge)
is_mother_of(Maggie, Bart)
is_mother_of(Maggie, Homer)
is_mother_of(Marge, Lisa)
is_mother_of(Marge, Maggie)

Inference: Forward Chaining

- Input: set of known atoms
  Input: set of Horn clauses
  Input: target formula to prove (or disprove)

  - Implication where all **antecedents** (i.e., conditions) and **consequent** (i.e., result) are positive

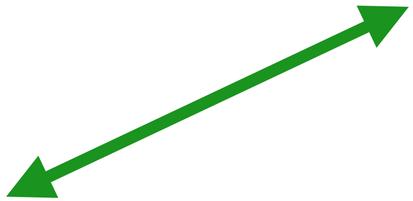
  - E.g., ~~∀x, ∀y,~~ is_mother_of(x, y) → is_woman(y)

  - Generate new knowledge using **generalized modus ponens**

Generalized Modus Ponens

E.g., is_mother_of(A, B) ∧ is_mother_of(C, B) → sibling(A, C)

$pred_1(v_{11}, \ldots) \wedge pred_2(v_{21}, \ldots) \wedge \ldots \wedge pred_n(v_{n1}, \ldots) \rightarrow pred_r(v_{r1}, \ldots)$



KB

$pred_1(A, \ldots)$
$pred_2(B, \ldots)$
$pred_1(C, \ldots)$
…

Substitution

$v_{11} = A$
$v_{21} = C$
$v_{22} = A$
$v_{31} = B$
…

Handling Quantifiers

- Universal (∀): replace with any constant during substitution
- Existential (∃): replace existential variable with new constant

Forward Chaining, High Level

- For each sentence **r** in KB,
  - For each substitution **s** that satisfies the conditions of **r**
    - Add the consequent of **r** with substitution **s** to KB
    - Check if target sentence is in KB or if contradiction

First Order Logic

- Terminology: predicates, arity, terms, functions, atoms
- Quantifiers: universal, existential
- Inference: differences from propositional logic
  - Requires handling quantifiers, finding substitutions