First-order Logic

CS 486/686

Sept 25, 2008

University of Waterloo

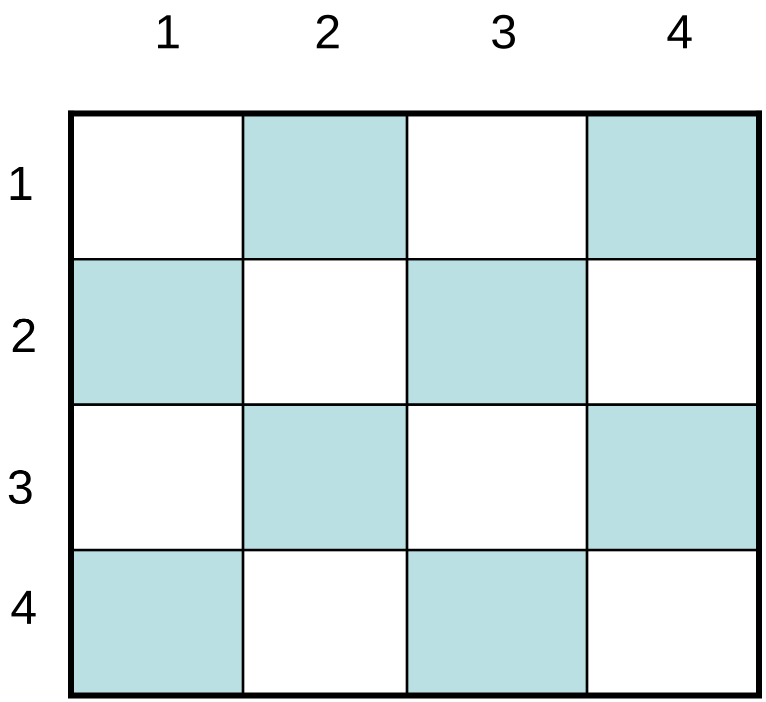
Outline

- First-order logic
  - Syntax and semantics
- Inference
  - Propositionalization with ground inference
  - Lifted resolution

Introduction

- Propositional logic provides a general language to encode deterministic domain knowledge.
- While the encoding is precise and well defined it is often tedious or complicated.
- Simple English sentences often lead to long logical formula.
- Can we make propositional logic more concise and more natural?

Recall: 4-Queens problem



- At least one queen in row 1:

$$Q_{11} \vee Q_{12} \vee Q_{13} \vee Q_{14}$$

- At most one queen in row 1:

$$Q_{11} \Rightarrow \neg Q_{12} \wedge \neg Q_{13} \wedge \neg Q_{14}$$
$$Q_{12} \Rightarrow \neg Q_{11} \wedge \neg Q_{13} \wedge \neg Q_{14}$$
$$Q_{13} \Rightarrow \neg Q_{11} \wedge \neg Q_{12} \wedge \neg Q_{14}$$
$$Q_{14} \Rightarrow \neg Q_{11} \wedge \neg Q_{12} \wedge \neg Q_{13}$$

Repeat for every row, column and diagonal.
This is too tedious! Can we be more concise?

First-order Logic

- World
  - consist of objects and relations
  - Has many objects
- First-order logic:
  - Natural: use predicates to encode relations and constants to encode objects
  - Concise: use quantifiers (e.g., $\forall$, $\exists$) to talk about many objects simultaneously

First-order Logic Syntax

- Sentence → Predicate(Term, …)
  | Term = Term
  | (Sentence Connective Sentence)
  | Quantifier Variable, … Sentence
  | ¬Sentence
- Term → Function(Term, …) | Constant | Variable
- Connective → ⇒ | ∧ | ∨ | ⇔
- Quantifier → ∀ | ∃
- Constant → A | X_1 | John | …
- Variable → a | x | s | …
- Predicate → Before | HasColor | Raining | …
- Function → Mother | LeftLeg | …

Example: kinship domain

- Elizabeth is the mother of Charles
  Mother(Elizabeth,Charles)
- Charles is the father of William
  Father(Charles,William)
- One's grandmother is the mother of one's parent
  $\forall x,z\ \exists y\ Grandmother(x,z) \Leftrightarrow Mother(x,y) \wedge Parent(y,z)$

Symbols

- Constant symbols: objects
  - E.g. William, Elizabeth, Charles
- Predicate symbols: relationships
  - Binary: Mother(,), Grandmother(,)
  - Unary: Female( )
  - Predicates have a truth value
- Function symbols: functions
  - Denote an object
  - E.g.: MotherOf(William) = Elizabeth

Quantifiers

- Universal quantifier: $\forall$
  - For all
  - $\forall x\ P(x) \equiv P(const1) \wedge P(const2) \wedge \ldots$
- Existential quantifier: $\exists$
  - There exists
  - $\exists x\ P(x) \equiv P(const1) \vee P(const2) \vee \ldots$

Nested Quantifiers

- Order of identical quantifiers doesn't matter
- Brothers are siblings
  - $\forall x\ \forall y\ Brother(x,y) \Rightarrow siblings(x,y)$
  - $\forall y\ \forall x\ Brother(x,y) \Rightarrow siblings(x,y)$
- Similarly for existential quantifiers
  - $\exists x\ \exists y\ P(x,y) \equiv \exists y\ \exists x\ P(x,y)$

Nested Quantifiers

- Order of different quantifiers matters
- $\forall x\ \exists y$ Loves(x,y)
  - Everyone loves someone
  - Conjunction of disjunctions (CNF)
- $\exists x\ \forall y$ Loves(x,y)
  - There is a person that loves everyone
  - Disjunction of conjunction (DNF)

Connections between $\forall$ and $\exists$

- We only need one of the quantifiers

- De Morgan's rule
  - $\forall x\ \neg P \equiv \neg\exists x\ P$ $\quad$ $\neg P \land \neg Q \equiv \neg(P \lor Q)$
  - $\neg\forall x\ P \equiv \exists x\ \neg P$ $\quad$ $\neg(P \land Q) \equiv \neg P \lor \neg Q$
  - $\forall x\ P \equiv \neg\exists x\ \neg P$ $\quad$ $P \land Q \equiv \neg(\neg P \lor \neg Q)$
  - $\exists x\ P \equiv \neg\forall x\ \neg P$ $\quad$ $P \lor Q \equiv \neg(\neg P \land \neg Q)$

Equality

- Special relation
- We could define an “Equality” predicate
  - $x = y \equiv Equality(x,y)$
    - True: when x and y are the same
    - False: otherwise

Practice: 4-Queens problem

- 4-queens problem in first-order logic
- Predicates:
  - Queen(,)
- Constants:
  - 1, 2, 3, 4 (column and row numbers)
- Column constraints:
  - $\forall i, j_1, j_2\ Queen(i, j_1) \wedge j_1 \neq j_2 \Rightarrow \neg Queen(i, j_2)$
  - $\forall i\ \exists j\ Queen(i, j)$

Practice: street puzzle

- Predicates:
  - House(), Person( ), Color( ), Drink( ), Job( ), Animal( )
  - Attr(,)   (attribute of)
- Constants:
  - 1, 2, 3, 4, 5 (house number)
  - English, Spaniard, Japanese, Italian, Norwegian
  - Red, Green, White, Yellow, Blue
  - Tea, Coffee, Milk, Juice, Water
  - Painter, Sculptor, Diplomat, Violinist, Doctor
  - Dog, Snails, Fox, Horse, Zebra
- Function:
  - Left(x): number of the house to the immediate left of x

Practice: street puzzle

- The Spaniard has a dog
  - $\forall x\ House(x) \Rightarrow (Attr(x,Spaniard) \Leftrightarrow Attr(x,Dog))$
- the green house is on the immediate left of the red house
  - $\neg Attr(1,Red) \wedge$
  - $\neg Attr(5,Green) \wedge$
  - $\forall x\ House(x) \wedge x \neq 1 \Rightarrow$
    $(Attr(x,Red) \Leftrightarrow Attr(Left(x),Green))$

Propositional vs first-order logic

- Propositional logic:
  - variables
- First-order logic:
  - Quantifiers, predicates, constants, functions

- Syntactically different!
- Are they equally expressive?
  - Prop logic → 1st-order logic: yes
  - 1st-order logic → prop logic: yes (finite domains)
    no (infinite domains)

Propositional → first-order logic

- Variables → predicates
- Indices → constants

- 4-queens problem:
  - $Q_{11} \Rightarrow \neg Q_{12} \wedge \neg Q_{13} \wedge \neg Q_{14}$
  - $Q(1,1) \Rightarrow \neg Q(1,2) \wedge \neg Q(1,3) \wedge \neg Q(1,4)$

First-order → propositional logic

- Quantifiers
  - ∀ → conjunction
  - ∃ → disjunction
- Predicates → variables
- Constants → indices
- Functions of constants → functions of indices

First-order → propositional logic

- $\forall x\ House(x) \wedge x \neq 1 \Rightarrow$
  $(Attr(x,Red) \Leftrightarrow Attr(Left(x),Green))$
- $\forall x\ House_x \wedge x \neq 1 \Rightarrow (Attr_{x,Red} \Leftrightarrow Attr_{Left(x),Green})$
- $(House_1 \wedge 1 \neq 1 \Rightarrow (Attr_{1,Red} \Leftrightarrow Attr_{Left(1),Green})) \wedge$
  $(House_2 \wedge 2 \neq 1 \Rightarrow (Attr_{2,Red} \Leftrightarrow Attr_{Left(2),Green})) \wedge$
  $(House_3 \wedge 3 \neq 1 \Rightarrow (Attr_{3,Red} \Leftrightarrow Attr_{Left(3),Green})) \wedge$
  $(House_4 \wedge 4 \neq 1 \Rightarrow (Attr_{4,Red} \Leftrightarrow Attr_{Left(4),Green})) \wedge$
  $(House_5 \wedge 5 \neq 1 \Rightarrow (Attr_{5,Red} \Leftrightarrow Attr_{Left(5),Green}))$

Inference

- Propositionalize KB
  - Run favorite ground inference algorithm (e.g., backtracking (DPLL), resolution
  - Efficient?
    - No: propositionalizing may yield an exponentially large formula
    - $\forall x\ \exists y\ \forall z\ P(x,y,z) \rightarrow$ conjunction of disjunctions of conjunctions
    - $O(n^m)$ where n is # of constants and m is # of quantifiers

- Alternative: lifted inference
  - Work directly with first-order formula

Lifted Inference

- First-order logic:
  - Quantifiers allow us to characterize several objects simultaneously
- Lifted inference:
  - Do inference by reasoning about several terms simultaneously
  - Ground terms only when necessary
  - Hope: determine truth value of goal formula without grounding all terms

Lifted Resolution

Two phases

1. Reduce to lifted conjunctive normal form
2. Resolution with unification

Reduction to Lifted CNF

Six steps

1. Eliminate implications
2. Move negations inwards
3. Standardize variables
4. Skolemize
5. Drop universal quantifiers
6. Distribute $\vee$ over $\wedge$

Reduction to Lifted CNF

Example:
$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$

1. Eliminate implications

   $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

2. Move negations inwards (apply DeMorgan's rules)

   $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$
   $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
   $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

Reduction to Lifted CNF

$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

3. Standardize variables (use different names)

   $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$

4. Skolemize (replace existential variables by new constants or functions)

   $\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

Skolemization

- $\exists x\ P(x)$
  - There is at least one term x that satisfies P(x)
  - We don't care what that term is
- Idea: create a new constant
  - Let *A* be a new constant which could correspond to any object, but since we don't care what that object is, just denote it by this new constant.
  - $\exists x\ P(x) \rightarrow P(A)$

Skolemization

- $\forall x \, \exists y \, P(x,y)$
  - For each x, there is at least one term y that satisfies P(x,y)
  - We don't care what that term is but it may be different for different x
- Idea: create a new function
  - Let f be a new function that denotes the satisfying term for each x
  - $\forall x \, \exists y \, P(x,y) \rightarrow P(x,f(x))$

Reduction to Lifted CNF

$\forall x$ [Animal(F(x)) $\wedge$ $\neg$Loves(x,F(x))] $\vee$ [Loves(G(x),x)]

5. Drop universal quantifiers (since all remaining variables are universally quantified)

   [Animal(F(x)) $\wedge$ $\neg$Loves(x,F(x))] $\vee$ [Loves(G(x),x)]

6. Distribute $\vee$ over $\wedge$ (to get a CNF)

   [Animal(F(x)) $\vee$ Loves(G(x),x)] $\wedge$
   [$\neg$Loves(x,F(x)) $\vee$ Loves(G(x),x)]

Resolution with Unification

- Recall
  - Unit resolution:
    - $A \wedge (A \Rightarrow B) \models B$
    - $A \wedge (\neg A \vee B) \models B$
  - General resolution:
    - $(A \Rightarrow B) \wedge (B \Rightarrow C) \models (A \Rightarrow C)$
    - $(\neg A \vee B) \wedge (\neg B \vee C) \models (\neg A \vee C)$
- Lifted resolution: terms may not match exactly because of variables

Unification

- Variables are like "wild cards" that can be anything

- Find unifier:
  - Unify Knows(John,x) and Knows(John,Jane)
    → Knows(John,Jane) {x/Jane}
  - Unify Knows(John,x) and Knows(y,Bill)
    → Knows(John,Bill) {x/Bill, y/John}
  - Unify Knows(John,x) and Knows(y,Mother(y))
    → Knows(John,Mother(John))
    {x/Mother(John), y/John}

Most General Unifier

- Find unifier that is as general as possible (i.e., preserves as many variables as possible)
- Find most general unifier:
  - Unify Knows(John,x) and Knows(y,Mother(z))
    → Knows(John,Mother(z))
    {y/John, x/Mother(z)}

Resolution with Unification

- Find clauses with positive and negative terms that can be unified
- Eliminate unifying terms and perform substitutions in remaining terms according to most general unifier

- Unit resolution:
  - $A(f(x),x)) \wedge [\neg A(y,g(z)) \vee B(y,z)] \models B(f(g(z)),z)$
  - $\{x/g(z),y/f(g(z))\}$
- General resolution:
  - $[\neg A(x,y) \vee B(y,f(z))] \wedge [\neg B(u,v) \vee C(v,w)] \models [\neg A(x,y) \vee C(f(z),w)]$
  - $\{u/y,v/f(z)\}$

Next class

- Reasoning under uncertainty
  - Russell and Norvig, Chapter 13