COMP4418: Knowledge Representation and Reasoning

First-Order Logic

Maurice Pagnucco
School of Computer Science and Engineering
University of New South Wales
NSW 2052, AUSTRALIA
morri@cse.unsw.edu.au
First-Order Logic

- First-order logic furnishes us with a much more expressive knowledge representation language than propositional logic

- We can directly talk about objects, their properties, relations between them, etc. . . .

- Here we discuss first-order logic and resolution

- However, there is a price to pay for this expressiveness in terms of decidability

References:

- Ivan Bratko, Prolog Programming for Artificial Intelligence, Addison-Wesley, 2001. (Chapter 15)

Overview

- Syntax of First-Order Logic
- Semantics of First-Order Logic
- Conjunctive Normal Form
- Unification
- First-Order Resolution
- Soundness and Completeness
- Decidability
- Conclusion
Syntax of First-Order Logic

■ Constant Symbols: \(a, b, \ldots, Mary\) (objects)
■ Variables: \(x, y, \ldots\)
■ Function Symbols: \(f, mother \_ of, sine, \ldots\)
■ Predicate Symbols: \(Mother, likes, \ldots\)
■ Quantifiers: \(\forall\) (universal); \(\exists\) (existential)

Terms: constant, variable, functions applied to terms (refer to objects)

Atomic Sentences: predicate applied to terms (state facts)

Ground (closed) term: a term with no variable symbols
Syntax of First-Order Logic

Sentence ::= AtomicSentence || Sentence Connective Sentence
       || Quantifier Variable Sentence || ~ Sentence || ( Sentence )

AtomicSentence ::= Predicate ( Term* )

Term ::= Function ( Term* ) || Constant || Variable

Connective ::= → || ∧ || ∨ || ↔

Quantifier ::= ∀ || ∃

Constant ::= a || John || . . .

Variable ::= x || men || . . .

Predicate ::= P || Red || Between || . . .

Function ::= f || Father || . . .
Converting English into First-Order Logic

- Everyone likes lying on the beach — $\forall x \ Beach(x)$
- Someone likes Fido — $\exists x \ Likes(x, Fido)$
- No one likes Fido — $\neg \exists x \ Likes(x, Fido)$
- Fido doesn’t like everyone — $\neg \forall x \ Likes(Fido, x)$
- All cats are mammals — $\forall x \ (Cat(x) \rightarrow Mammal(x))$
- Some mammals are carnivorous — $\exists x \ (Mammal(x) \land Carnivorous(x))$
Nested Quantifiers

Note that the order of quantification is very important

- Everything likes everything — $\forall x \ \forall y \ Likes(x, \ y)$
- Something likes something — $\exists x \ \exists y \ Likes(x, \ y)$
- Everything likes something — $\forall x \ \exists y \ Likes(x, \ y)$
- There is something liked by everything — $\exists y \ \forall x \ Likes(x, \ y)$
Scope of Quantifiers

- The scope of a quantifier in a formula $\phi$ is that subformula $\psi$ of $\phi$ of which that quantifier is the main logical operator.
- Variables belong to the innermost quantifier that mentions them.
- Examples:
  - $Q(x) \rightarrow \forall y P(x, y)$ — scope of $\forall y$ is $P(x, y)$
  - $\forall z P(z) \rightarrow \neg Q(z)$ — scope of $\forall z$ is $P(z)$ but not $Q(z)$
  - $\exists x (P(x) \rightarrow \forall x P(x))$
  - $\forall x (P(x) \rightarrow Q(x)) \rightarrow (\forall x P(x) \rightarrow \forall x Q(x))$
**Terminology**

- **Free-variable occurrences in a formula** —
  - All variables in an atomic formula
  - The free-variable occurrences in $\neg \phi$ are those in $\phi$
  - The free-variable occurrences in $\phi \oplus \psi$ are those in $\phi$ and $\psi$ for any connective $\oplus$
  - The free-variable occurrences in $\forall x \Phi$ and $\exists x \Phi$ are those in $\Phi$ except for occurrences of $x$

- **Open formula** — A formula in which free variables occur

- **Closed formula** — A formula with no free variables

- Closed formulae are also known as sentences
Semantics of First-Order Logic

- A world in which a sentence is true under a particular interpretation is known as a model of that sentence under the interpretation.

- **Constant symbols** an interpretation specifies which object in the world a constant refers to

- **Predicate symbols** an interpretation specifies which relation in the model a predicate refers to

- **Function symbols** an interpretation specifies which function in the model a function symbol refers to

- **Universal quantifier** is true iff all instances are true

- **Existential quantifier** is true iff one instance is true
Conversion into Conjunctive Normal Form

1. Eliminate implication

\[ \phi \rightarrow \psi \equiv \neg \phi \lor \psi \]

2. Move negation inwards (negation normal form)

\[
\neg (\phi \land \psi) \equiv \neg \phi \lor \neg \psi
\]
\[
\neg (\phi \lor \psi) \equiv \neg \phi \land \neg \psi
\]
\[
\neg \forall x \phi \equiv \exists x \neg \phi
\]
\[
\neg \exists x \phi \equiv \forall x \neg \phi
\]
\[
\neg \neg \phi \equiv \phi
\]

3. Standardise variables

\[
(\forall x P(x)) \lor (\exists x Q(x))
\]

becomes

\[
(\forall x P(x)) \lor (\exists y Q(y))
\]
Conversion into Conjunctive Normal Form

4. Skolemise

\[ \exists x \, P(x) \Rightarrow P(a) \]
\[ \forall x \exists y \, P(x, y) \Rightarrow \forall x \, P(x, f(x)) \]
\[ \forall x \forall y \exists z \, P(x, y, z) \Rightarrow \forall x \forall y \, P(x, y, f(x, y)) \]

5. Drop universal quantifiers

6. Distribute \( \land \) over \( \lor \)

\[ (\phi \land \psi) \lor \chi \equiv (\phi \lor \chi) \land (\psi \lor \chi) \]

7. Flatten nested conjunctions and disjunctions

\[ (\phi \land \psi) \land \chi \equiv \phi \land \psi \land \chi; (\phi \lor \psi) \lor \chi \equiv \phi \lor \psi \lor \chi \]

(8. In proofs, rename variables in separate clauses — standardise apart)
CNF — Example 1

∀x[∀y P(x, y) → ¬∀y(Q(x, y) → R(x, y))]

1. ∀x[¬(∀y P(x, y)) ∨ ¬∀y(¬Q(x, y) ∨ R(x, y))]

2. ∀x[(∃y P(x, y)) ∨ ∃y(Q(x, y) ∧ ¬R(x, y))]

3. ∀x[(∃y ¬P(x, y)) ∨ ∃z(Q(x, z) ∧ ¬R(x, z))]

4. ∀x[¬P(x, f(x)) ∨ (Q(x, g(x)) ∧ ¬R(x, g(x)))]

5. ¬P(x, f(x)) ∨ (Q(x, g(x)) ∧ ¬R(x, g(x))]

6. (¬P(x, f(x)) ∨ Q(x, g(x))) ∧ (¬P(x, f(x)) ∨ ¬R(x, g(x)))

8. ¬P(x, f(x)) ∨ Q(x, g(x))

¬P(y, f(y)) ∨ ¬R(y, g(y))
CNF — Example 2

\[\neg \exists x \forall y \forall z ((P(y) \lor Q(z)) \rightarrow (P(x) \lor Q(x)))\]
\[\neg \exists x \forall y \forall z (\neg (P(y) \lor Q(z)) \lor (P(x) \lor Q(x))) \quad \text{[Eliminate } \rightarrow ]\]
\[\forall x \neg \forall y \forall z (\neg (P(y) \lor Q(z)) \lor (P(x) \lor Q(x))) \quad \text{[Move } \neg \text{ inwards]}\]
\[\forall x \exists y \neg \forall z (\neg (P(y) \lor Q(z)) \lor (P(x) \lor Q(x))) \quad \text{[Move } \neg \text{ inwards]}\]
\[\forall x \exists y \exists z (\neg (P(y) \lor Q(z)) \lor (P(x) \lor Q(x))) \quad \text{[Move } \neg \text{ inwards]}\]
\[\forall x \exists y \exists z (\neg \neg (P(y) \lor Q(z)) \land \neg (P(x) \lor Q(x))) \quad \text{[Move } \neg \text{ inwards]}\]
\[\forall x \exists y \exists z (((P(y) \lor Q(z)) \land \neg (P(x) \land \neg Q(x)))) \quad \text{[Move } \neg \text{ inwards]}\]
\[\forall x ((P(f(x)) \lor Q((g(x)))) \land \neg (P(x) \land \neg Q(x))) \quad \text{[Skolemise]}\]
\[(P(f(x)) \lor Q((g(x)))) \land \neg P(x) \land \neg Q(x) \quad \text{[Drop } \forall ]\]
Unification

- Unification takes two atomic formulae and returns a substitution that makes them look the same
- Example:
  \[ \{ x/a, y/z, w/f(b, c) \} \]
- Note:
  1. Each variable has at most one associated expression
  2. No variable with an associated expression occurs within any associated expression
- \( \{ x/g(y), y/f(x) \} \) is not a substitution
- Substitution \( \sigma \) that makes a set of expressions identical known as a unifier
- Substitution \( \sigma_1 \) is a more general unifier than a substitution \( \sigma_2 \) if for some substitution \( \tau \), \( \sigma_2 = \sigma_1 \tau \). 

© UNSW, 2019
Generated: 15 September 2019
First-Order Resolution

- Generalised Resolution Rule:

  For clauses $\chi \lor \Phi$ and $\neg \Psi \lor \zeta$

  $$\begin{array}{c}
  \chi \lor \Phi \\
  \neg \Psi \lor \zeta \\
  \hline
  (\chi \lor \zeta).\theta
  \end{array}$$

- Where $\theta$ is a unifier for atomic formulae $\Phi$ and $\Psi$

- $\chi \lor \zeta$ is known as the resolvent
Resolution — Example 1

\[\forall x(P(x) \rightarrow \forall x \neg P(x))\]

\[\neg \exists x(P(x) \rightarrow \forall x \neg P(x))\]

\[\forall x(\neg P(x) \lor \forall x P(x)) \quad \text{[Drive \ \neg \ \text{inwards}]}\]

\[\forall x(\neg \neg P(x) \land \neg \forall x P(x)) \quad \text{[Drive \ \neg \ \text{inwards}]}\]

\[\forall x(P(x) \land \exists x \neg P(x)) \quad \text{[Drive \ \neg \ \text{inwards}]}\]

\[\forall x(P(x) \land \exists z \neg P(z)) \quad \text{[Standardise Variables]}\]

\[\forall x(P(x) \land \neg P(f(x))) \quad \text{[Skolemise]}\]

\[P(x) \land \neg P(f(x)) \quad \text{[Drop \ \forall]}\]

1. \(P(x)\) \quad \text{[\neg \ \text{Conclusion}]}\]

2. \(\neg P(f(y))\) \quad \text{[\neg \ \text{Conclusion}]}\]

3. \(P(f(y))\) \quad \{1. \{x/f(y)\}\}\]

4. \(\Box\) \quad \{2, 3. \text{Resolution}\}
Resolution — Example 2

1. \( P(f(x)) \lor Q(g(x)) \) \[\neg\) Conclusion\]
2. \( \neg P(y) \) \[\neg\) Conclusion\]
3. \( \neg Q(z) \) \[\neg\) Conclusion\]
4. \( P(f(a)) \lor Q(g(a)) \) \[1. \{x/a\}\]
5. \( \neg P(f(a)) \) \[2. \{y/f(a)\}\]
6. \( \neg Q(g(a)) \) \[3. \{z/g(a)\}\]
7. \( Q(g(a)) \) \[4, 5. Resolution\]
8. \( \Box \) \[6, 7. Resolution\]
Resolution — Example 3

1. \(\text{man}(\text{Marcus})\)  \([\text{Premise}]\)
2. \(\text{Pompeian}(\text{Marcus})\)  \([\text{Premise}]\)
3. \(\neg \text{Pompeian}(x) \lor \text{Roman}(x)\)  \([\text{Premise}]\)
4. \(\text{ruler}(\text{Caesar})\)  \([\text{Premise}]\)
5. \(\neg \text{Roman}(y) \lor \text{loyaltyto}(y, \text{Caesar}) \lor \text{hate}(y, \text{Caesar})\)  \([\text{Premise}]\)
6. \(\text{loyaltyto}(z, f(z))\)  \([\text{Premise}]\)
7. \(\neg \text{man}(w) \lor \neg \text{ruler}(u) \lor \neg \text{tryassassinate}(w, u) \lor \neg \text{loyaltyto}(w, u)\)  \([\text{Premise}]\)
8. \(\text{tryassassinate}(\text{Marcus}, \text{Caesar})\)  \([\text{Premise}]\)
9. \(\neg \text{hate}(\text{Marcus}, \text{Caesar})\)  \([\neg \text{Conclusion}]\)
10. \(\neg \text{Roman}(\text{Marcus}) \lor \text{loyaltyto}(\text{Marcus}, \text{Caesar}) \lor \neg \text{hate}(\text{Marcus}, \text{Caesar})\)  \([5. \{y/\text{Marcus}\}]\)
11. \(\neg \text{Roman}(\text{Marcus}) \lor \text{loyaltyto}(\text{Marcus}, \text{Caesar})\)  \([9, 10. \text{Resolution}]\)
Resolution — Example 3

12. \(\neg\)Pompeian(Marcus) \(\lor\) Roman(Marcus) \hspace{1em} [3. \{x/Marcus\}]
13. loyaltyto(Marcus, Caesar) \(\lor\) \(\neg\)Pompeian(Marcus) \hspace{1em} [11, 12. Resolution]
14. loyaltyto(Marcus, Caesar) \hspace{1em} [2, 13. Resolution]
15. \(\neg\)man(Marcus) \(\lor\) \(\neg\)ruler(Caesar) \(\lor\) \(\neg\)tryassassinate(Marcus, Caesar) \(\lor\) \(\neg\)loyaltyto(Marcus, Caesar) \hspace{1em} [7. \{w/Marcus, u/Caesar\}]
16. \(\neg\)man(Marcus) \(\lor\) \(\neg\)ruler(Caesar) \(\lor\) \(\neg\)tryassassinate(Marcus, Caesar) \hspace{1em} [14, 15. Resolution]
17. \(\neg\)ruler(Caesar) \(\lor\) \(\neg\)tryassassinate(Marcus, Caesar) \hspace{1em} [1, 16. Resolution]
18. \(\neg\)tryassassinate(Marcus, Caesar) \hspace{1em} [4, 17. Resolution]
19. \(\Box\) \hspace{1em} [8, 18. Resolution]
Soundness and Completeness

Resolution is
- sound (if $\lambda \vdash \rho$, then $\lambda \models \rho$)
- complete (if $\lambda \models \rho$, then $\lambda \vdash \rho$)

Decidability

- First-order logic is not decidable
- How would you prove this?
Conclusion

- First-order logic allows us to speak about objects, properties of objects and relationships between objects.
- It also allows quantification over variables.
- First-order logic is quite an expressive knowledge representation language; much more so than propositional logic.
- However, we do need to add things like equality if we wish to be able to do things like counting.
- We have also traded expressiveness for decidability.
- How much of a problem is this?
- If we add (Peano) axioms for mathematics, then we encounter Gödel’s famous incompleteness theorem (which is beyond the scope of this course).