

COMP9414: Artificial Intelligence

First-Order Logic

Wayne Wobcke

Room J17-433

wobcke@cse.unsw.edu.au

Based on slides by Maurice Pagnucco

Overview

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

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)
 - ▶ Stuart J. Russell and Peter Norvig, [Artificial Intelligence: A Modern Approach](#), Third Edition, Pearson Education, 2010. (Chapters 8, 9)

Syntax of First-Order Logic

- **Constant symbols:** $a, b, \dots, Mary$ (objects)
- **Variables:** x, y, \dots
- **Function symbols:** $f, mother_of, sine, \dots$
- **Predicate symbols:** $Mother, likes, \dots$
- **Quantifiers:** \forall (universal); \exists (existential)

Language of First-Order Logic

- **Terms:** constants, variables, functions applied to terms (refer to objects)
 - ▶ e.g. a , $f(a)$, $mother_of(Mary)$, ...
- **Atomic formulae:** predicates applied to tuples of terms
 - ▶ e.g. $likes(Mary, mother_of(Mary))$, $likes(x, a)$
- **Quantified formulae:**
 - ▶ e.g. $\forall x likes(x, a)$, $\exists x likes(x, mother_of(y))$
 - ▶ here the second occurrences of x are **bound** by the quantifier and y is **free**

Nested Quantifiers

The order of quantification is very important

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

Converting English into First-Order Logic

- Everyone likes lying on the beach — $\forall x likes_lying_on_beach(x)$
- Someone likes Fido — $\exists x likes(x, Fido)$
- No one likes Fido — $\neg \exists x likes(x, Fido)$ (or $\forall x \neg likes(x, Fido)$)
- 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 — $\exists x (mammal(x) \wedge carnivorous(x))$
- Note: $\forall x A(x) \equiv \neg \exists x \neg A(x)$, $\exists x A(x) \equiv \neg \forall x \neg A(x)$

Scope of Quantifiers

- The **scope** of a quantifier in a formula A is that subformula B of A 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 $\forall y P(x, y)$
 - ▶ $\forall z P(z) \rightarrow \neg Q(z)$ — scope of $\forall z$ is $\forall z P(z)$ but not $Q(z)$
 - ▶ $\exists x (P(x) \rightarrow \forall x P(x))$
 - ▶ $\forall x (P(x) \rightarrow Q(x)) \rightarrow (\forall x P(x) \rightarrow \forall x Q(x))$

Semantics of First-Order Logic

- An **interpretation** is required to give semantics to first-order logic. The interpretation is a non-empty “domain of discourse” (set of objects). The truth of any formula depends on the interpretation.
- The interpretation provides, for each:
 - constant symbol** an object in the domain
 - function symbols** a function from domain tuples to the domain
 - predicate symbol** a relation over the domain (a set of tuples)
- Then we define:
 - universal quantifier** $\forall x P(x)$ is True iff $P(a)$ is True for all assignments of domain elements a to x
 - existential quantifier** $\exists x P(x)$ is True iff $P(a)$ is True for at least one assignment of domain element a to x

Conversion to Conjunctive Normal Form

1. Eliminate implications and bi-implications as in propositional case
2. Move negations inward using De Morgan’s laws
 - ▶ plus rewriting $\neg\forall x P$ as $\exists x \neg P$ and $\neg\exists x P$ as $\forall x \neg P$
3. Eliminate double negations
4. Rename bound variables if necessary so each only occurs once
 - ▶ e.g. $\forall x P(x) \vee \exists x Q(x)$ becomes $\forall x P(x) \vee \exists y Q(y)$
5. Use equivalences to move quantifiers to the left
 - ▶ e.g. $\forall x P(x) \wedge Q$ becomes $\forall x (P(x) \wedge Q)$ where x is not in Q
 - ▶ e.g. $\forall x P(x) \wedge \exists y Q(y)$ becomes $\forall x \exists y (P(x) \wedge Q(y))$

Towards Resolution for First-Order Logic

- Based on resolution for propositional logic
- Extended syntax: allow variables and quantifiers
- Define “clausal form” for first-order logic formulae
- Eliminate quantifiers from clausal forms
- Adapt resolution procedure to cope with variables (unification)

Conversion to CNF — Continued

6. Skolemise (replace each existentially quantified variable by a **new term**)
 - ▶ $\exists x P(x)$ becomes $P(a_0)$ using a Skolem constant a_0 since $\exists x$ occurs at the outermost level
 - ▶ $\forall x \exists y P(x, y)$ becomes $P(x, f_0(x))$ using a Skolem function f_0 since $\exists y$ occurs within $\forall x$
7. The formula now has only universal quantifiers and all are at the left of the formula: drop them
8. Use distribution laws to get CNF and then clausal form

CNF — Example 1

$$\forall x [\forall y P(x, y) \rightarrow \neg \forall y (Q(x, y) \rightarrow R(x, y))]$$

1. $\forall x [\neg \forall y P(x, y) \vee \neg \forall y (\neg Q(x, y) \vee R(x, y))]$
- 2, 3. $\forall x [\exists y \neg P(x, y) \vee \exists y (Q(x, y) \wedge \neg R(x, y))]$
4. $\forall x [\exists y \neg P(x, y) \vee \exists z (Q(x, z) \wedge \neg R(x, z))]$
5. $\forall x \exists y \exists z [\neg P(x, y) \vee (Q(x, z) \wedge \neg R(x, z))]$
6. $\forall x [\neg P(x, f(x)) \vee (Q(x, g(x)) \wedge \neg R(x, g(x)))]$
7. $\neg P(x, f(x)) \vee (Q(x, g(x)) \wedge \neg R(x, g(x)))$
8. $(\neg P(x, f(x)) \vee Q(x, g(x))) \wedge (\neg P(x, f(x)) \vee \neg R(x, g(x)))$
8. $\{\neg P(x, f(x)) \vee Q(x, g(x)), \neg P(x, f(x)) \vee \neg R(x, g(x))\}$

CNF — Example 2

$$\neg \exists x \forall y \forall z ((P(y) \vee Q(z)) \rightarrow (P(x) \vee Q(x)))$$

1. $\neg \exists x \forall y \forall z (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$
2. $\forall x \neg \forall y \forall z (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$
2. $\forall x \exists y \neg \forall z (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$
2. $\forall x \exists y \exists z \neg (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$
2. $\forall x \exists y \exists z ((P(y) \vee Q(z)) \wedge \neg (P(x) \vee Q(x)))$
6. $\forall x ((P(f(x)) \vee Q(g(x))) \wedge \neg P(x) \wedge \neg Q(x))$
7. $(P(f(x)) \vee Q(g(x)) \wedge \neg P(x) \wedge \neg Q(x))$
8. $\{P(f(x)) \vee Q(g(x)), \neg P(x), \neg Q(x)\}$

Unification

- A **unifier** of two atomic formulae is a **substitution** of terms **for variables** that makes them identical
 - ▶ Each variable has at most one associated term
 - ▶ Substitutions are applied simultaneously
- Unifier of $P(x, f(a), z)$ and $P(z, z, u) : \{x/f(a), z/f(a), u/f(a)\}$
- Substitution σ_1 is a **more general unifier** than a substitution σ_2 if for some substitution τ , $\sigma_2 = \sigma_1 \tau$ (i.e. σ_1 followed by τ)
- **Theorem.** If two atomic formulae are unifiable, they have a most general unifier

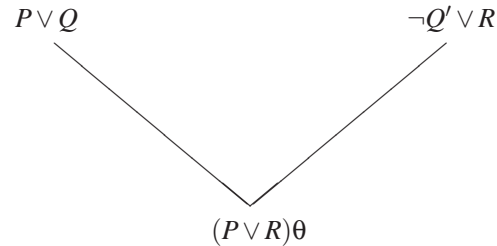
Examples

- $\{P(x, a), P(b, c)\}$ is not unifiable
- $\{P(f(x), y), P(a, w)\}$ is not unifiable
- $\{P(x, c), P(b, c)\}$ is unifiable by $\{x/b\}$
- $\{P(f(x), y), P(f(a), w)\}$ is unifiable by $\sigma = \{x/a, y/w\}$, $\tau = \{x/a, y/a, w/a\}$, $\nu = \{x/a, y/b, w/b\}$
Note that σ is an m.g.u. and $\tau = \sigma\theta$ where $\theta = \dots$?
- $\{P(x), P(f(x))\}$ is not unifiable (c.f. occur check!)

First-Order Resolution

Generalised Resolution Rule

For clauses $P \vee Q$ and $\neg Q' \vee R$ with Q, Q' atomic formulae



- where θ is a most general unifier for Q and Q'
- $(P \vee R)\theta$ is the **resolvent** of the two clauses

Applying Resolution Refutation

- Negate query to be proven (resolution is a refutation system)
- Convert knowledge base and negated query into CNF and extract clauses
- Repeatedly apply resolution to clauses **or copies of clauses** until either the empty clause (contradiction) is derived or no more clauses can be derived (a copy of a clause is the clause with all variables renamed)
- If the empty clause is derived, answer ‘yes’ (query follows from knowledge base), otherwise answer ‘no’ (query does not follow from knowledge base)

Resolution — Example 1

$$\vdash \exists x (P(x) \rightarrow \forall x P(x))$$

$$\text{CNF}(\neg \exists x (P(x) \rightarrow \forall x P(x)))$$

$$1, 2. \forall x \neg(\neg P(x) \vee \forall x P(x))$$

$$2. \forall x (\neg \neg P(x) \wedge \neg \forall x P(x))$$

$$2, 3. \forall x (P(x) \wedge \exists x \neg P(x))$$

$$4. \forall x (P(x) \wedge \exists y \neg P(y))$$

$$5. \forall x \exists y (P(x) \wedge \neg P(y))$$

$$6. \forall x (P(x) \wedge \neg P(f(x)))$$

$$8. P(x), \neg P(f(x))$$

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

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

$$3. \square \quad [1, 2 \text{ Resolution } \{x/f(y)\}]$$

Resolution — Example 2

$$\vdash \exists x \forall y \forall z ((P(y) \vee Q(z)) \rightarrow (P(x) \vee Q(x)))$$

$$1. P(f(x)) \vee Q(g(x)) \quad [\neg \text{Conclusion}]$$

$$2. \neg P(x) \quad [\neg \text{Conclusion}]$$

$$3. \neg Q(x) \quad [\neg \text{Conclusion}]$$

$$4. \neg P(y) \quad [\text{Copy of 2}]$$

$$5. Q(g(x)) \quad [1, 4 \text{ Resolution } \{y/f(x)\}]$$

$$6. \neg Q(z) \quad [\text{Copy of 3}]$$

$$7. \square \quad [5, 6 \text{ Resolution } \{z/g(x)\}]$$

Soundness and Completeness Again

- First-order resolution refutation is **sound**, i.e. it preserves truth (if a set of premises are all true, any conclusion drawn from those premises **must** also be true)
- First-order resolution refutation is **complete**, i.e. it is capable of proving all consequences of any knowledge base (not shown here!)
- First-order resolution refutation is **not decidable**, i.e. there is **no** algorithm implementing resolution which when asked whether $S \vdash P$, can always answer 'yes' or 'no' (correctly)

Unification Algorithm

- The disagreement set of S : Find the leftmost position at which not all members E of S have the same symbol; the set of subexpressions of each E in S that begin at this position is the disagreement set of S .
- Algorithm
 - $S_0 = S, \sigma_0 = \{\}, i = 0$
 - If S_i is not a singleton find its disagreement set D_i , otherwise terminate with σ_i as the most general unifier
 - If D_i contains a variable v_i and term t_i such that v_i does not occur in t_i then

$$\sigma_{i+1} = \sigma_i\{v_i/t_i\}, S_{i+1} = S_i\{v_i/t_i\}$$
 otherwise terminate as S is not unifiable
 - $i = i + 1$; resume from step 2

Examples

- $S = \{f(x, g(x)), f(h(y), g(h(z)))\}$
 $D_0 = \{x, h(y)\}$ so $\sigma_1 = \{x/h(y)\}$
 $S_1 = \{f(h(y), g(h(y))), f(h(y), g(h(z)))\}$
 $D_1 = \{y, z\}$ so $\sigma_2 = \{x/h(z), y/z\}$
 $S_2 = \{f(h(z), g(h(z))), f(h(z), g(h(z)))\}$
 i.e. σ_2 is an m.g.u.
- $S = \{f(h(x), g(x)), f(g(x), h(x))\} \dots?$

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 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)