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:
Language of First-Order Logic

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

Converting English into First-Order Logic

- Everyone likes lying on the beach — $\forall x \ \text{likes}(\text{lying}_o n_{\text{beach}}(x))$
- Someone likes Fido — $\exists x \ \text{likes}(x, \text{Fido})$
- No one likes Fido — $\neg \exists x \ \text{likes}(x, \text{Fido})$ (or $\forall x \ \neg \text{likes}(x, \text{Fido})$)
- Fido doesn’t like everyone — $\neg \forall x \ \text{likes}(\text{Fido}, x)$
- All cats are mammals — $\forall x (\text{cat}(x) \rightarrow \text{mammal}(x))$
- Some mammals are carnivorous — $\exists x (\text{mammal}(x) \land \text{carnivorous}(x))$
- Note: $\forall A(x) \equiv \neg \exists x \ \neg A(x)$, $\exists x A(x) \equiv \neg \forall x \ \neg A(x)$

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 $x$

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 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. $\forall x P(x) \lor \exists y Q(x)$ becomes $\forall x (P(x) \lor \exists y Q(y))$
5. Use equivalences to move quantifiers to the left
   - e.g. $\forall x P(x) \land Q$ becomes $\forall x (P(x) \land Q)$ where $x$ is not in $Q$
   - e.g. $\forall x P(x) \land \exists y Q(y)$ becomes $\forall x \exists y (P(x) \land Q(y))$
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**

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

1. \[ \forall x [\forall y P(x, y) \lor \forall y (\neg Q(x, y) \lor R(x, y))] \]
2. \[ \forall x [\exists y \neg P(x, y) \lor \exists y (Q(x, y) \land \neg R(x, y))] \]
3. \[ \forall x [\exists y \exists z [\neg P(x, y) \lor (Q(x, z) \land \neg R(x, z))] \]
4. \[ \forall x [\exists y \exists z [\neg P(x, y) \lor (Q(x, z) \land \neg R(x, z))] \]
5. \[ \forall x \exists y \exists z [\neg P(x, y) \lor (Q(x, z) \land \neg R(x, z))] \]
6. \[ \forall x [\neg P(x, f(x)) \lor (Q(x, g(x)) \land \neg R(x, g(x)))] \]
7. \[ \forall x [\neg P(x, f(x)) \lor (Q(x, g(x)) \land \neg R(x, g(x)))] \]
8. \[ \forall x [\neg P(x, f(x)) \lor (Q(x, g(x)) \land \neg R(x, g(x)))] \]

**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 \( \sigma_1 \) is a more general unifier than a substitution \( \sigma_2 \) if for some substitution \( \tau \), \( \sigma_2 = \sigma_1 \tau \) (i.e. \( \sigma_1 \) followed by \( \tau \))

**Theorem.** If two atomic formulae are unifiable, they have a most general unifier

**CNF — Example 2**

\[ \neg \exists x \forall y \forall z ((P(y) \lor Q(z)) \rightarrow (P(x) \lor Q(x))) \]

1. \[ \neg \exists x \forall y \forall z ((P(y) \lor Q(z)) \lor P(x) \lor Q(x)) \]
2. \[ \forall x \neg \forall y \forall z ((P(y) \lor Q(z)) \lor P(x) \lor Q(x)) \]
3. \[ \forall x \exists y \exists z ((P(y) \lor Q(z)) \lor P(x) \lor Q(x)) \]
4. \[ \forall x \exists y \exists z ((P(y) \lor Q(z)) \lor P(x) \lor Q(x)) \]
5. \[ \forall x \exists y \exists z ((P(y) \lor Q(z)) \lor P(x) \lor Q(x)) \]
6. \[ \forall x ((P(f(x)) \lor Q(g(x))) \land \neg P(x) \land \neg Q(x)) \]
7. \[ \forall x ((P(f(x)) \lor Q(g(x))) \land \neg P(x) \land \neg Q(x)) \]
8. \[ \forall x ((P(f(x)) \lor Q(g(x))) \land \neg P(x) \land \neg Q(x)) \]

**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
  - \( \sigma = \{ x/a, y/w \}, \tau = \{ x/a, y/a, w/a \}, u = \{ x/a, y/b, w/b \} \)
  - Note that \( \sigma \) is an m.g.u. and \( \tau = \sigma \theta \) where \( \theta = \ldots ? \)
- \( \{ P(x), P(f(x)) \} \) is not unifiable (c.f. occur check!)
First-Order Resolution

Generalised Resolution Rule

For clauses $P ∨ Q$ and $¬Q' ∨ R$ with $Q, Q'$ atomic formulae

$$(P ∨ R)\theta$$

where $\theta$ is a most general unifier for $Q$ and $Q'$

$(P ∨ R)\theta$ 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

\[ \exists x (P(x) → ∀x P(x)) \]

CNF($\neg\exists x (P(x) → ∀x P(x))$)

1. $∀x \neg P(x) → ∀x P(x)$
2. $∀x (\neg P(x) ∧ ∀x P(x))$
3. $∀x (P(x) ∧ ∃x \neg P(x))$
4. $∀x (P(x) ∧ ∃y \neg P(y))$
5. $∀x (P(x) ∧ ∃y \neg P(y))$
6. $∀x (P(x) ∧ ∃y \neg P(y))$
7. $P(x)$
8. $P(x)$

1. $P(x)$ \[ ¬ Conclusion \]
2. $\neg P(f(y))$ \[ Copy of $\neg$ Conclusion \]
3. $\Box$ \[ 1, 2 Resolution \{x/f(y)\} \]

Resolution — Example 2

\[ \exists x ∀y ∀z ((P(y) ∨ Q(z)) → (P(x) ∨ Q(x))) \]

1. $P(f(x)) ∨ Q(g(x))$ \[ ¬ Conclusion \]
2. $\neg P(x)$ \[ ¬ Conclusion \]
3. $\neg Q(x)$ \[ ¬ Conclusion \]
4. $\neg P(y)$ \[ Copy of 2 \]
5. $Q(g(x))$ \[ 1, 4 Resolution \{y/f(x)\} \]
6. $\neg Q(z)$ \[ Copy of 3 \]
7. $\Box$ \[ 5, 6 Resolution \{z/g(x)\} \]
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**
    1. $S_0 = S, \sigma_0 = \{}$, $i = 0$
    2. If $S_i$ is not a singleton find its disagreement set $D_i$, otherwise terminate with $\sigma_i$ as the most general unifier
    3. If $D_i$ contains a variable $v_i$ and term $t_i$ such that $v_i$ does not occur in $t_i$ then
       $$\sigma_{i+1} = \sigma_i[v_i/t_i], \quad S_{i+1} = S_i[v_i/t_i]$$
       otherwise terminate as $S$ is not unifiable
    4. $i = i + 1$; resume from step 2

Examples

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

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).