First-Order Logic

In a nutshell

- First-order logic furnishes us with a much more *expressive* language than propositional logic.
- We can directly talk about objects, their properties, relations between them, etc.
- However, there is a price to pay for this expressiveness in terms of *decidability*

Syntax

- Constant symbols: *a*, *b*,…, *Mary* (objects)
- Variables: *x*, *y*, . . .
- Function Symbols: *f*, *mother_of*, *sine*,…
- Predicate Symbols: *Mother*, *Likes*,…
- Quantifiers: ∀ (universal), ∃ (existential)

Language of first-order logic

- Terms: constants, variables, functions applied to terms (refer to *objects*)
  - e.g. *a*, *x*, *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. ∀*x Likes*(*x*, *a*), ∃*x Likes*(*x*, *mother_of*(*y*))
  - here the second occurrences of *x* are bound by the quantifier and *y* is free

Converting English into Logic

- Everyone likes lying on the beach — ∀*x Likes_lying_on_beach*(*x*)
- Someone likes Fido —∃*x Likes*(*x*, *Fido*)
- No one likes Fido — ¬∃*x Likes*(x, *Fido*) (or ∀*x*¬*Likes*(*x*, *Fido*))
- Fido doesn't like everyone — ¬∀*x Likes*(*Fido*, *x*)
- All cats are mammals — ∀*x* (*Cat*(*x*) → *Mammal*(*x*))
- Some mammals are carnivorous — ∃*x* (*Mammal*(*x*) ∧ *Carnivorous*(*x*))

Nested Quantifiers

The order of quantification is very important.

- Everyone likes everyone — ∀*x* ∀*y Likes*(*x*, *y*) (or ∀*y* ∀*x Likes*(*x*, *y*))
- Someone likes someone — ∃*x* ∃*y Likes*(*x*, *y*) (or ∃*y* ∃*x Likes*(*x*, *y*))
- Everyone likes someone — ∀*x* ∃*y Likes*(*x*, *y*)
- There is someone liked by everyone — ∃*y* ∀*x Likes*(*x*, *y*)

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 $Q(z)$
  - $\exists x\, (P(x) \rightarrow \forall x\, P(x))$ — scope of $\exists x$ is *not* in $\forall x\, P(x)$
  - $\forall x\, (P(x) \rightarrow Q(x)) \rightarrow (\forall x\, P(x) \rightarrow \forall x\, Q(x))$

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). It is also called as the "univeral set". The truth of any formula depends on the interpretation.
- The interpretation provides, for each:
  - constant symbol an object in the domain
  - function symbol 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*
- Note that all variables represent objects from the *same* domain of discourse.

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

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 $\forall x\, P(x) \vee \exists y\, Q(y)$
   - e.g. $\forall x\, (P(x) \wedge Q(x))$ becomes $\forall x\, P(x) \wedge \forall y\, Q(y)$
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 $\forall x\, \exists y\, (P(x) \wedge Q(y))$

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* $\forall x$
   - $\forall v \forall w \exists x \exists y\, P(v,w,x,y)$ *becomes* $P(v,w, f_0(v,w), f_1(v,w))$
7. Drop universal quantifiers – these are all at the left and there are no existential quantifiers now. However, the dropped quantifiers are still implicitly present.
8. Use distribution laws to get CNF and clausal form.

CNF — Example 1

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

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

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

Unification

- A unifier of two atomic formulae is a substitution that makes them identical
  - Each variable has at most one associated expression
  - Apply substitutions simultaneously
- Unifier of $P(x,f(a),z)$ and $P(z,z,u)$:
  $\{\ x\ /\ f(a),\ z\ /\ f(a),\ u\ /\ f(a)\ \}$
- Substitution S_1 is a more general unifier than a substitution S_2 if for some substitution t, $S_2 = S_1\ t$ (i.e. S_1 followed by t).
- Theorem: If two atomic formulae are unifiable, they have a most general unifier.

First-Order Resolution

$P \vee Q$ $\qquad\qquad$ $\neg Q' \vee R$

$(P \vee R)\theta$

- Here Q and Q' are atomic formulae
- 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 conclusion to be proven (resolution is a refutation system).
- Convert knowledge base and negated conclusion 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

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

Resolution — Example 1

$\Rightarrow \exists x\,(P(x) \rightarrow \forall x\, P(x))$

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

Resolution — Example 2

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

Resolution Strategies

- Unit Preference: Prefer statements containing single literals. This will produce *shorter* statements
- Set of Support: Start with clauses from the negated conclusion as the "set of support". Combine them with premises. Put resulting statements in the set of support. Repeat.

Soundness and Completeness

- 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 – Godel's Completeness Theorem. (If $S \Rightarrow P$ is valid, it can be proved.)
- First-order resolution refutation is *not* decidable, i.e. there is *no* algorithm implementing resolution which when asked whether $S \Rightarrow$ P, can always answer 'yes' or 'no' (correctly).

Undecidability of 1st Order Logic

$P(f(x)) \rightarrow P(x) \Rightarrow P(a)$

- Obviously, this cannot be proved.
- However, let us attempt to show this using resolution:

$\neg P(f(x)) \vee P(x)$ $\quad\quad$ $\neg P(a)$

$\downarrow x / a$

$\neg P(f(a))$

$\downarrow x / f(a)$

$\neg P(f(f(a)))$

$\vdots$

Undecidability (contd)

- First-order logic is complete, so if an argument is valid, it can be proved using resolution.
- But if argument is invalid, then the search tree will not contain the empty clause and the search *may* go on forever.
- Even in the propositional case (which is decidable), complexity of resolution is $O(2^n)$ for problems of size n

Horn Clauses

Idea: Use less expressive language

- Review
  - literal — atomic formula or negation of atomic formula
  - clause — disjunction of literals
- Definite Clause – exactly one positive literal
  - e.g. $C \vee \neg A_1 \vee \dots \vee \neg A_n$, i.e. $C \leftarrow A_1 \wedge \dots \wedge A_n$ (Prolog rule)
- Negative Clause – no positive literals
  - e.g. $\neg Q$ (negation of a query)
- Horn Clause – clause with at most one positive literal
- If first-order logic is limited to horn clauses, it is still *undecidable*!
- However, resolution in propositional logic become efficient – can be solved in polynomial time

Equality in First-Order Logic

- E.g. *Brother*(*A*) = *B* is a statement
- Handling equality in inference: demodulation rule

$$x = y, (\dots z \dots) \Rightarrow (\dots y \dots)$$

- Here (…*z*…) is any sentence containing *z* where *z* unifies with *x*
- Then, replace *z* with *y*

Higher-Order Logic

- In first-order logic we can quantify over *objects* but not over relations or functions.
- Higher-order logic allows us to quantify over relations and functions
- Example:
  $\forall x,y\ (x = y) \leftrightarrow (\forall p\ p(x) \leftrightarrow p(y))$
- Higher-order logics have strictly more expressive power than first-order logic.

Peano's Axioms

- Based on 3 concepts:
  - A constant: *zero*
  - A predicate indicating numbers: *N*
  - A successor function: *S*
- Axioms:
  - $N(zero)$ [i.e. zero is a number]
  - $\forall x\ N(x) \rightarrow N(S(x))$
  - $\forall x,y\ [\ N(x) \wedge N(y) \wedge S(x) = S(y) \rightarrow x = y\ ]$
  - $\neg\ (\exists x\ N(x) \wedge (S(x) = zero))$
  - $\forall \varphi\ [\ \varphi(zero) \wedge \forall x\ (\ N(x) \wedge (\varphi(x) \rightarrow \varphi(S(x)))) \rightarrow$
    $\forall x\ (\ N(x) \rightarrow \varphi(x)\ )\ ]$
- These axioms incorporate mathematical induction
- Incorporating peano's axioms in 1^{st} order logic makes it incomplete! – Godel's incompleteness theorem

Conclusion

- First-order logic is an expressive formal language and allows for powerful reasoning
- Think in terms of having millions of statements in a knowledge base, rather than the few premises you see in class assignments
- Theorem proving is undecidable in general. However, this should not deter us from applying first-order logic where-ever possible.