Artificial Intelligence

First-order logic

Conjunctive normal form

Resolution theorem-proving

Limitations of Propositional Logic

- Many kinds of inference cannot be formalized in propositional logic. For example, most useful inferences involve applying a general rule to a specific case. But general-to-specific inferences like the following cannot be formalized in propositional logic.
  - All men are mortal
  - Socrates is a man
  - Therefore, Socrates is mortal
- This inference cannot be formalized in propositional logic because it refers to individual men, such as Socrates, and make generalizations about all men.

First-Order Logic (FOL)

- Propositional logic has only sentences, which represent facts.
- First-order logic extends propositional logic in two directions
  - It provides an inner structure for sentences. They are viewed as expressing relations between objects or individuals.
  - It provides a means to express, and reason with, generalizations. It makes it possible to say that a certain property holds of all objects, of some objects, or of no object.

Terms

- First-order logic has terms that represent objects or individuals. Terms are built using constant, variable, and function symbols.
  - **Constants** (designate specific object): A, B, John, Red, etc.
  - **Variables** (designate unspecified object): x, y, z, etc.
  - **Functions** (designate a specific object related in a certain way to another object, or objects): FatherOf, ColorOf, etc.
- Examples of terms:
  - ColorOf(CarOf(Bill)), CostOf(TextbookOf(CMPSCI383)), etc.
- **Ground terms** are terms that include no variables.

Predicates

- Predicates have a value of true or false
- A predicate can take arguments, which are terms
- A predicate with one argument expresses a property of an object
  - Student(Bob)
- A predicate with two or more arguments expresses a relation between objects
  - likes(Bob, Mary)
  - likes(Bob, school-of (Bob))
- A predicate with no arguments is a simple proposition, as in propositional logic

Universal Quantifier

$\forall$x P(x) means "for all x, P of x is true"

Example: $\forall$x Happy(x)

If the universe of discourse is people, then this means that everyone is happy.

Other examples:

$\forall x\ \forall y$ Knows(x,y) => Knows(y,x)

$\forall x\ \forall y$ Knows(x,y) ^ Knows(y,x)

$\forall x\ \forall y$ Knows(x,y) => ¬ Likes(y,x)

Existential Quantifier

∃x P(x) means "there exists at least one x for which P of x is true"

Example: ∃x Happy(x)

If the universe of discourse is people, then this means there is at least one happy person.

Other examples:

∀x ∃y Knows(x,y)

∃x ∃y Knows(x,y) ^ Knows(y,x)

∀x ∃y Knows(x,y) => ¬ Likes(y,x)

Relationship between Universal and Existential Quantifiers

∀x ¬ P(x)	<=>	¬∃x P(x)
¬∀x P(x)	<=>	∃x ¬P(x)
∀x P(x)	<=>	¬∃x ¬ P(x)
∃x P(x)	<=>	¬∀x ¬ P(x)

Examples:

¬∃y Happy(y)	<=>	∀y ¬ Happy(y)
∃y ¬Happy(y)	<=>	¬ ∀y Happy(y)
∀x ¬ Likes(x,Jane)	<=>	¬∃x Likes(x,Jane)
∀x Likes(x,Jane)	<=>	¬∃x ¬Likes(x,Jane)

Scope of Quantifiers

- The scope of a quantifier extends throughout the expression unless the same variable is re-quantified.
- Examples:

  ∀x ∃y¬ Likes(x,y) ∧ ∃y Married(x,y)

  ∀x ∃y¬ Likes(x,y)∧ ∃z Married(x,z)

  ∀x ∃y ∃z ¬ Likes(x,y) ∧ Married(x,z)

  ∀x ∃y¬ Likes(x,y) ∧ Married(x,y)

Order matters: Some examples

- ∀x ∃y Likes(x,y)
- ∃y ∀x Likes(x,y)
- ∀x ∃y Married(x,y)
- ∃y ∀x Married(x,y)
- ¬∀x ∃y Married(x,y)
- ¬∃y ∀x Married(x,y)

Translate from English to FOL

- All people like dogs.
- Some people like cats.
- Some dogs do not like any cats.
- People who like cats are nice but are not happy.
- Cats and dogs are both types of animals.
- My cat named Fred likes a dog named Dolly but does not like a dog named Max.
- Happy people are not sad.

Translate from English to FOL

- All CS students are smart.
- Some CS students study music.
- Not all CS students study music.
- Everybody has a mother.

Translate from English to FOL

- You can fool some of the people all of the time.
- You can't fool all of the people all of the time.
- There is one and only one Pope.

The Semantics of FOL

- An interpretation is a set of objects and a mapping from each constant, functional symbol and predicate to an appropriate object, function and relation in the domain.
- A sentence can be valid, satisfiable, or unsatisfiable, just as in propositional logic.

Inference in First-Order Logic

- Godel (1930) proved that FOL is complete, which means that a proof can be found for any valid sentence
- However, it was not until 1965 that Robinson developed the first *algorithm* for finding proofs in FOL, called resolution refutation
- (Truth tables don't work in FOL because sentences with variables can have an infinite number of possible interpretations.)

Resolution

- Assumes normal form
- Using conjunctive normal form
  From $(A \vee B) \wedge (\neg A \vee C)$ infer $(A \vee C)$
- Using implicative normal form
  From $(\neg A \Rightarrow B) \wedge (B \Rightarrow C)$ infer $(\neg A \Rightarrow C)$

Resolution Refutation

To prove that a sentence p can be derived from a set of sentences KB:
- Convert ¬p and the sentences in KB to CNF.
- Repeat until the empty clause results (a contradiction) or no clauses can be resolved
  - Find two clauses to which the resolution rule applies, but has not previously been applied.
  - Apply the resolution rule to create a new clause.
- If terminate with empty clause, p is proved. Otherwise, p cannot be proved.

Resolution in FOL

- To generalize resolution refutation from propositional logic to FOL, we have to answer two questions:
  - How do we convert a sentence to normal form (for example, CNF) when it contains quantifiers?
  - How do we detect that two literals contradict each other (and so can be resolved) when they contain variables?
- The answer to the 2nd question involves the concept of "unification," discovered by Robinson.
- Conversion to normal form and unificationare the two complicated aspects of resolution theorem-proving in FOL, and so we must discuss them in detail

Normal Form

- It is easier for a computer to perform logical inference if we convert all sentences into a "normal form." Any sentence of first-order logic can be transformed into an equivalent sentence in either of these normal forms.
- In **conjunctive normal form** (CNF), every sentence is expressed as a conjunction of clauses (where a clause is a disjunction of literals)
  $(P(x) \vee R(x,y)) \wedge (\neg S(y) \vee R(y,z)) \wedge T(y)$
- In **implicative normal form**, every sentence is expressed as an implication with a conjunction of atoms on the left and a disjunction of atoms on the right
  $P(x) \wedge Q(x,y) \Rightarrow S(y) \vee T(x,y)$

Converting to CNF

1) Eliminate the $\Leftrightarrow$ connective using the equivalence between $P \Leftrightarrow Q$ and $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$
2) Eliminate the $\Rightarrow$ connective using the equivalence between $P \Rightarrow Q$ and $\neg P \vee Q$
3) Move negation symbols inward, using the following equivalences:
   - $\neg(P \vee Q)$ becomes $\neg P \wedge \neg Q$
   - $\neg(P \wedge Q)$ becomes $\neg P \vee \neg Q$
   - $\neg \forall x P$ becomes $\exists x \neg P$
   - $\neg \exists x P$ becomes $\forall x \neg P$
   - $\neg \neg P$ becomes P

Converting to CNF (continued)

4) Standardize variable names so that each quantifier has its own unique variable name.
   $\forall x P \vee \exists y Q$ becomes $\forall x \exists y\, P \vee Q$
5) Move all quantifiers to the left of the sentence (preserving their order).
   $\forall x P \vee \exists y Q$ becomes $\forall x \exists y\, P \vee Q$
6) Use **skolemization** to remove existential quantifiers (see next slide)
7) Drop universal quantifier symbols (since we now assume all variables are universally quantified)

Converting to CNF (continued)

8) Use distributive law to convert to CNF
   $(P \wedge Q) \vee R$ becomes $(P \vee R) \wedge (Q \vee R)$
9) Flatten nested conjunctions and disjunctions
   $(P \vee Q) \vee R$ becomes $(P \vee Q \vee R)$

Skolemization

- Method for converting a sentence with existential quantifiers into a sentence without existential quantifiers such that the first sentence is satisfiable if and only if the second is.
- To eliminate an existential quantifier, replace each occurrence of its variable by a **Skolem function** whose arguments are the variables of universal quantifiers whose scope includes the scope of the existential quantifier being eliminated.
- If the existential quantifier being eliminated is not within the scope of any universal quantifiers, the Skolem function has no arguments, that is, it is a constant.

Skolemization: Example

$\forall x \exists y\, (Person(x) \wedge Person(y)) \Rightarrow Loves(x,y)$

is converted to

$\forall x\, (Person(x) \wedge Person(f(x)) \Rightarrow Loves(x,f(x))$

where f(x) specifies the person that x loves

Skolemization: Another Example

- The sentence “Everyone has a brain” is represented as
  - $\forall x\ Person(x) \Rightarrow \exists y\ Brain(y) \wedge Has(x,y)$
- If we simply substitute the constant B for the existentially quantified variable y, we get a sentence that says “Everyone has the same brain,” not what we want to say!
  - $\forall x\ Person(x) \Rightarrow Brain(B) \wedge Has(x,B)$
- Using the Skolem function B(x) to represent a function that denotes the object that is x’s brain, the correct skolemization of our original sentence is
  - $\forall x\ Person(x) \Rightarrow Brain(B(x)) \wedge Has(x,B(x))$

Skolemization: More examples

$\exists x P(x)$ becomes $P(A)$

$\forall x \forall y \exists z P(x,y,z)$ becomes $\forall x \forall y P(x,y,F(x,y))$

$\forall x \exists y Pred(x,y)$ becomes $\forall x Pred(x,Succ(x))$

Practice

Convert the following sentences to CNF

$\forall x\ P(x) \Rightarrow \quad \forall y \exists x\ Q(y,x)$

Conversion to Implicative Normal Form

- To convert to implicative normal form, first convert to CNF, then perform one additional step.
- From each conjunct, build an equivalent implication by putting each negative literal on the left hand side and each positive literal on the right
- $(\neg A \vee \neg B \vee C \vee D)$ becomes $(A \wedge B) \Rightarrow (C \vee D)$

Unification

- In propositional logic, it is easy to see that two literals (such as p and ¬p) contradict each other.
- In FOL, this matching process is more complicated because arguments of predicates must be considered. For example, man(John) and ¬man(John) is a contradiction, while man(John) and man(Spot) is not.
- To detect contradictions in FOL, we need a matching procedure that compares two literals and discovers whether there exists a set of **substitutions** that makes them identical. This procedure is called unification.
- Notation: x/car means car is substituted for x

Unification cont.

- The unification algorithm takes two atomic sentences (literals), such as Knows(John, x) and Knows(John, Paul), and returns a substitution that makes them look the same, such as {x/Paul}
- If there is no such substitition, the unification fails.

Most General Unifier

- There may be more than one substitution that unifies two clauses. In fact, there may be infinitely many
- The unification algorithm returns the "most general unifier", that is, the substitution that makes the least commitment about the bindings of variables

Example

Rule:
Knows(John, x) => Hates(John, x)
Knowledge Base:
Knows (John, Jane)
Knows(y,Leonid)
Knows(y, Mother-of(y))
Knows(x, Elizabeth)

Unify(Knows(John,x), Knows(John,Jane)) =
Unify(Knows(John, x), Knows(y, Leonid)) =
Unify(Knows(John,x), Knows(y, Mother(y))) =
Unify(Knows(John,x), Knows(x,Elizabeth)) =

Standardizing Variables Apart

Rule:
Knows(John, x_1) => Hates(John, x_1)
Knowledge Base:
Knows (John, Jane)
Knows(x_2 ,Leonid)
Knows(x_3 , Mother-of(x_3))
Knows(x_4 , Elizabeth)

Unify(Knows(John, x_1), Knows (x_2,Elizabeth)) =

Resolution Example

Anyone passing his history exams and winning the lottery is happy. But anyone who studies or is lucky can pass all his exams. John did not study but John is lucky. Anyone who is lucky wins the lottery. Is John happy?

1st step: Convert to predicate logic

1. Anyone passing his history exams and winning the lottery is happy.
   $\forall x$ Pass(x, History) $\wedge$ Win(x, Lottery) $\Rightarrow$ Happy(x)
2. But anyone who studies or is lucky can pass all his exams.
   $\forall x$ $\forall y$ Study(x) $\vee$ Lucky(x) $\Rightarrow$ Pass(x,y)
3. John did not study, but John is lucky
   $\neg$ Study(John) $\wedge$ Lucky(John)
4. Anyone who is lucky wins the lottery.
   $\forall x$ Lucky(x) $\Rightarrow$ Win(x, Lottery)

2nd step: Convert to CNF

Eliminate implications:

1. $\forall x$ $\neg$ (Pass(x, History) $\wedge$ Win(x, Lottery)) $\vee$ Happy(x)
2. $\forall x$ $\forall y$ $\neg$ (Study(x) $\vee$ Lucky(x)) $\vee$ Pass(x,y)
3. $\neg$ Study(John) $\wedge$ Lucky(John)
4. $\forall x$ $\neg$ Lucky(x) $\vee$ Win(x, Lottery)

Convert to CNF Cont.

Move ¬ inward

1. ∀x ¬ Pass(x, History) ∨ ¬ Win(x, Lottery)) ∨ Happy(x)
2. ∀x ∀y (¬ Study(x) ∧ ¬Lucky(x))∨ Pass(x,y)
3. ¬ Study(John) ∧ Lucky(John)
4. ∀x ¬ Lucky(x) ∨ Win(x, Lottery)

Standardize variables: no action needed

Move quantifiers left: no action needed except drop quantifiers

Skolemize: no action needed

Convert to CNF Cont.

Distribute ∧ over ∨

1. ¬ Pass(x, History) ∨ ¬ Win(x, Lottery)) ∨ Happy(x)
2. (¬ Study(x) ∨ Pass(x,y)) ∧ (¬ Lucky(x) ∨ Pass(x,y))
3. ¬ Study(John) ∧ Lucky(John)
4. ¬ Lucky(x) ∨ Win(x, Lottery)

Convert to CNF Cont.

Flatten nested conjunctions and disjunctions
no action necessary

State as a set of disjunction of literals

1. ¬ Pass(x, History) ∨ ¬ Win(x, Lottery) ∨ Happy(x)
2. a. ¬ Study(x) ∨ Pass(x,y)
2. b. ¬Lucky(x) ∨ Pass(x,y)
3. a. ¬ Study(John)
   b. Lucky(John)
4. ¬ Lucky(x) ∨ Win(x, Lottery)

Convert to CNF Cont.

Standardize variables apart

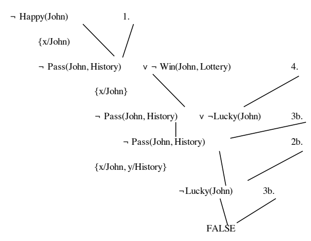
1. ¬ Pass(x1, History) ∨ ¬ Win(x1, Lottery) ∨ Happy(x1)
2. a. ¬ Study(x2) ∨ Pass(x2,y1)
2. b. ¬Lucky(x3) ∨ Pass(x3,y2)
3. a. ¬ Study(John)
   b. Lucky(John)
4. ¬ Lucky(x4) ∨ Win(x4, Lottery)

NOW IN CONJUNCTIVE NORMAL FORM (CNF)

3rd step: Resolution Proof Procedure

- Assert negation of goal
  - In this case the goal is to prove
    Happy(John)
  - Add the clause
    ¬ Happy(John)
    to the KB
- Resolve clauses together until FALSE is derived

Resolution Proof Tree


¬ Happy(John)
1.
{x/John)
¬ Pass(John, History) ∨ ¬ Win(John, Lottery)
4.
{x/John}
¬ Pass(John, History) ∨ ¬Lucky(John)
3b.
¬ Pass(John, History)
2b.
{x/John, y/History}
¬Lucky(John)
3b.
FALSE