Artificial Intelligence

Inference in First-Order 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
- Conjunctive normal form
  From $(A \vee B) \wedge (\neg A \vee C)$ infer $(A \vee C)$
- 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 $\neg p$ 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 clause form (CNF) when it contains quantifiers?
- How do we detect that two literals contradict each other (and so can be resolved) when they contain variables?

Conjunctive Normal Form

- Every sentence is a conjunction of disjunctions of literals
- Can convert any FOL KB into CNF

(1) **Remove implications**

Replace $P \Rightarrow Q$ by $\neg P \vee Q$

Replace $P \Leftrightarrow Q$ by $(\neg P \vee Q) \wedge (P \vee \neg Q)$

Conjunctive Normal Form cont.

(2) **Move negation inwards**

$\neg\forall x\, P$ becomes $\exists x\, \neg P$

$\neg\exists x\, P$ becomes $\forall x\, \neg P$

$\neg\neg P$ becomes P

$\neg(P \wedge Q)$ is replaced by $\neg P \vee \neg Q$

$\neg(P \vee Q)$ is replaced by $\neg P \wedge \neg Q$

(3) **Standardize variables**

each quantifier gets unique variables

e.g. $\exists x P(x) \wedge \exists x Q(x)$ becomes $\exists x P(x) \wedge \exists y Q(y)$

Conjunctive Normal Form cont.

(4) **Move quantifiers to the left**

$\forall x P \vee \exists y Q$ becomes $\forall x \exists y\, P \vee Q$

(5) **Eliminate $\exists$ by Skolemization.**

$\exists x P(x)$ 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 $\forall x Pred(x,Succ(x))$

(6) **Drop universal quantifiers**

(7) **Distribute And over Or**

$(P \wedge Q) \vee R$ becomes $(P \vee R) \wedge (Q \vee R)$

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 determine that two literals contradict each other. Simply look for p and ¬p.
- 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 return a substitution that makes them look the same, such as {x/Paul}
- If there is no such substitition, the unification fails.
- See pp. 302-3 for pseudocode

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 from pp. 270-271

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)) =

CS 4633/6633 Artificial Intelligence

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?

CS 4633/6633 Artificial Intelligence

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)

CS 4633/6633 Artificial Intelligence

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)

CS 4633/6633 Artificial Intelligence

Convert to CNF Cont.

Move $\neg$ inward

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

Standardize variables: no action needed

Move quantifiers left: no action needed except drop quantifiers

Skolemize: no action needed

CS 4633/6633 Artificial Intelligence

Convert to CNF Cont.

Distribute $\wedge$ over $\vee$

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

Convert to CNF Cont.

Flatten nested conjunctions and disjunctions
no action necessary
State as a set of disjunction of literals

1. $\neg$ Pass(x, History) $\vee$ $\neg$ Win(x, Lottery) $\vee$ Happy(x)

2. a. $\neg$ Study(x) $\vee$ Pass(x,y)

2. b. $\neg$Lucky(x) $\vee$ Pass(x,y)

3. a. $\neg$ Study(John)

   b. Lucky(John)

4. $\neg$ Lucky(x) $\vee$ Win(x, Lottery)

CS 4633/6633 Artificial Intelligence

Convert to CNF Cont.

Standardize variables apart

1. $\neg$ Pass(x1, History) $\vee$ $\neg$ Win(x1, Lottery) $\vee$ Happy(x1)

2. a. $\neg$ Study(x2) $\vee$ Pass(x2,y1)

2. b. $\neg$Lucky(x3) $\vee$ Pass(x3,y2)

3. a. $\neg$ Study(John)

   b. Lucky(John)

4. $\neg$ Lucky(x4) $\vee$ Win(x4, Lottery)

NOW IN CONJUNCTIVE NORMAL FORM (CNF)

CS 4633/6633 Artificial Intelligence

Resolution Proof Procedure

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

CS 4633/6633 Artificial Intelligence