Chapter 9 Inference in First-Order Logic

CS4811 - Artificial Intelligence

Nilufer Onder
Department of Computer Science
Michigan Technological University

Outline

Reducing first-order inference to propositional inference
- Universal instantiation
- Existential instantiation

Unification

Resolution

Universal instantiation (UI)

Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v\, \alpha}{\textsc{Subst}(\{v/g\}, \alpha)}$$

for any variable v and ground term g.

E.g., $\forall x King(x) \wedge Greedy(x) \implies Evil(x)$ yields

$$\begin{array}{l} King(John) \wedge Greedy(John) \implies Evil(John) \\ King(Richard) \wedge Greedy(Richard) \implies Evil(Richard) \\ King(Father(John)) \wedge Greedy(Father(John)) \implies Evil(Father(John)) \\ \vdots \end{array}$$

Existential instantiation (EI)

For any sentence α, variable v, and constant symbol k *that does not appear elsewhere in the knowledge base*:

$$\frac{\exists v\, \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

E.g., $\exists x Crown(x) \wedge OnHead(x, John)$ yields

$$Crown(C_1) \wedge OnHead(C_1, John)$$

provided C_1 is a new constant symbol, called a *Skolem constant*.

Another example: from $\exists x\, d(x^y)/dy = x^y$ we obtain

$$d(e^y)/dy = e^y$$

provided e is a new constant symbol.

Instantiation

- Universal instantiation can be applied several times to **add** new sentences:
  the new KB is logically equivalent to the old.
- Existential instantiation can be applied once to **replace** the existential sentence:
  the new KB is **not** equivalent to the old,
  but is satisfiable iff the old KB was satisfiable.

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x\, King(x) \land Greedy(x) \implies Evil(x)$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

Instantiating the universal sentence in **all possible** ways, we have

$King(John) \land Greedy(John) \implies Evil(John)$
$King(Richard) \land Greedy(Richard) \implies Evil(Richard)$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

The new KB is *propositionalized*: the proposition symbols are $King(John)$, $Greedy(John)$, $Evil(John)$, $King(Richard)$ etc.

Reduction (cont'd.)

- Claim: a *ground sentence* is entailed by new KB iff entailed by original KB.
- Claim: every FOL KB can be propositionalized so as to preserve entailment.
- Idea: propositionalize KB and query, apply resolution, return result.

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.
  E.g., from
  $\forall x King(x) \wedge Greedy(x) \implies Evil(x)$
  $King(John)$
  $\forall y Greedy(y)$
  $Brother(Richard, John)$

  it seems obvious that $Evil(John)$, but propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant.

  With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations.

  With function symbols, it gets much worse!

Problems with propositionalization (cont'd)

- With function symbols, there are infinitely many ground terms, e.g., *Father*(*Father*(*Father*(*John*))).
- Theorem: Herbrand (1930).
  If a sentence α is entailed by an FOL KB,
  it is entailed by a **finite** subset of the propositional KB.
- Idea: For $n = 0$ to ∞ do
  create a propositional KB by instantiating with depth-n terms
  see if α is entailed by this KB.
- Problem: works if α is entailed, loops if α is not entailed.
- Theorem: Entailment in FOL is *semidecidable*.
  Turing (1936), Church (1936)

Unification

We can get the inference immediately if we can find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$.

$\theta = \{x/John, y/John\}$ works

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, SteveJobs)$	$\{x/SteveJobs, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, SteveJobs)$	$fail$

Standardizing variables apart

- *Standardizing apart* eliminates overlap of variables.
- Rename all variables so that variables bound by different quantifiers have unique names.
- For example

  $$\forall x\, Apple(x) \implies Fruit(x)$$
  $$\forall x\, Spider(x) \implies Arachnid(x)$$

  is the same as

  $$\forall x\, Apple(x) \implies Fruit(x)$$
  $$\forall y\, Spider(y) \implies Arachnid(y)$$

Resolution

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Ken)\end{array}}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$.

Resolution refutation

- The general technique is to add the negation of the sentence to be proven to the KB and see if this leads to a contradiction.
- Idea: if the KB becomes inconsistent with the addition of the negated sentence, then the original sentence must be true.
- This is called *resolution refutation*.
- The procedure is complete for FOL.

Resolution refutation algorithm

function $\textsc{Resolution-Refutation}$ (KB, α)
returns true if KB $\models \alpha$

inputs:
KB, a knowledge base in CNF
α, a sentence in CNF

repeat
find two sentences s_1, s_2 to resolve
if not found **then return** false
$s_3 \leftarrow \textsc{Resolve}(s_1, s_2)$
if s_3 is the null clause
then return true
else $KB \leftarrow \cup s_3$

Conversion to CNF

1. Eliminate biconditionals and implications.
2. Reduce the scope of $\neg$: move $\neg$ inwards.
3. Standardize variables apart: each quantifier should use a different variable name.
4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a *Skolem function* of the enclosing universally quantified variables.
5. Drop all universal quantifiers: It's allright to do so now.
6. Distribute $\wedge$ over $\vee$.
7. Make each conjuct a separate clause.
8. Standardize the variables apart again.

Example 1

- All people who are graduating are happy.
  All happy people smile.
  JohnDoe is graduating.

  Is JohnDoe smiling?
- First convert to predicate logic
  $\forall x\, graduating(x) \implies happy(x)$
  $\forall x\, happy(x) \implies smiling(x)$
  $graduating(JohnDoe)$

  $smiling(JohnDoe)$ negate this: $\neg smiling(JohnDoe)$
- Then convert to canonical form.

Example 1 (cont'd)

1. $\forall x \, graduating(x) \implies happy(x)$
2. $\forall x \, happy(x) \implies smiling(x)$
3. $graduating(JohnDoe)$

4. $\neg smiling(JohnDoe)$

Step 1. Eliminate $\implies$

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall x \, \neg happy(x) \vee smiling(x)$
3. $graduating(JohnDoe)$
4. $\neg smiling(JohnDoe)$

Example 1 (cont'd)

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall x \, \neg happy(x) \vee smiling(x)$
3. $graduating(JohnDoe)$
4. $\neg smiling(JohnDoe)$

Step 2. Move $\neg$ inwards. (not needed)

Step 3. Standardize variables apart.

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall y \, \neg happy(y) \vee smiling(y)$
3. $graduating(JohnDoe)$
4. $\neg smiling(JohnDoe)$

Example 1 (cont'd)

1. $\forall x\, \neg graduating(x) \vee happy(x)$
2. $\forall y\, \neg happy(y) \vee smiling(y)$
3. $graduating(JohnDoe)$
4. $\neg smiling(JohnDoe)$

Step 4. Skolemize. (not needed)

Step 5. Drop all $\forall$.

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(JohnDoe)$
4. $\neg smiling(JohnDoe)$

Example 1 (cont'd)

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(JohnDoe)$
4. $\neg smiling(JohnDoe)$

Step 6. Distribute $\wedge$ over $\vee$. (not needed)

Step 7. Make each conjuct a separate clause. (not needed)

Step 8. Standardize the variables apart again. (not needed)

Ready for resolution!

Example 1 (cont'd)

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(JohnDoe)$
4. $\neg smiling(JohnDoe)$

Resolve 4 and 2 using $\theta = \{y/JohnDoe\}$:

5. $\neg happy(JohnDoe)$

Resolve 5 and 1 using $\theta = \{x/JohnDoe\}$:

6. $\neg graduating(JohnDoe)$

Resolve 6 and 3:

7. $\perp$

Example 2: Proving an existentially quantified sentence

- All people who are graduating are happy.
  All happy people smile.
  Someone is graduating.

  Is someone smiling?
- First convert to predicate logic
  $\forall x \, graduating(x) \implies happy(x)$
  $\forall x \, happy(x) \implies smiling(x)$
  $\exists x \, graduating(x)$

  $\exists x \, smiling(x)$ negate this: $\neg \exists x \, smiling(x)$
- Then convert to canonical form.

Example 2 (cont'd)

1. $\forall x\, graduating(x) \implies happy(x)$
2. $\forall x\, happy(x) \implies smiling(x)$
3. $\exists x\, graduating(x)$
4. $\neg\exists x\, smiling(x)$

Step 1. Eliminate $\implies$

1. $\forall x\, \neg graduating(x) \vee happy(x)$
2. $\forall x\, \neg happy(x) \vee smiling(x)$
3. $\exists x\, graduating(x)$
4. $\neg\exists x\, smiling(x)$

Example 2 (cont'd)

1. $\forall x\, \neg graduating(x) \lor happy(x)$
2. $\forall x\, \neg happy(x) \lor smiling(x)$
3. $\exists x\, graduating(x)$
4. $\neg \exists x\, smiling(x)$

Step 2. Move $\neg$ inwards.

1. $\forall x\, \neg graduating(x) \lor happy(x)$
2. $\forall x\, \neg happy(x) \lor smiling(x)$
3. $\exists x\, graduating(x)$
4. $\forall x\, \neg smiling(x)$

Example 2 (cont'd)

1. $\forall x\, \neg graduating(x) \vee happy(x)$
2. $\forall x\, \neg happy(x) \vee smiling(x)$
3. $\exists x\, graduating(x)$
4. $\forall x\, \neg smiling(x)$

Step 3. Standardize variables apart.

1. $\forall x\, \neg graduating(x) \vee happy(x)$
2. $\forall y\, \neg happy(y) \vee smiling(y)$
3. $\exists z\, graduating(z)$
4. $\forall w\, \neg smiling(w)$

Example 2 (cont'd)

1. $\forall x \neg graduating(x) \vee happy(x)$
2. $\forall y \neg happy(y) \vee smiling(y)$
3. $\exists z\, graduating(z)$
4. $\forall w \neg smiling(w)$

Step 4. Skolemize.

1. $\forall x \neg graduating(x) \vee happy(x)$
2. $\forall y \neg happy(y) \vee smiling(y)$
3. $graduating(NoName1)$
4. $\forall w \neg smiling(w)$

Example 2 (cont'd)

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall y \, \neg happy(y) \vee smiling(y)$
3. $graduating(NoName1)$
4. $\forall w \, \neg smiling(w)$

Step 5. Drop all $\forall$.

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(NoName1)$
4. $\neg smiling(w)$

Example 2 (cont'd)

1. $\neg graduating(x) \lor happy(x)$
2. $\neg happy(y) \lor smiling(y)$
3. $graduating(NoName1)$
4. $\neg smiling(w)$

Step 6. Distribute $\land$ over $\lor$. (not needed)

Step 7. Make each conjuct a separate clause. (not needed)

Step 8. Standardize the variables apart again. (not needed)

Ready for resolution!

Example 2 (cont'd)

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(NoName1)$
4. $\neg smiling(w)$

Resolve 4 and 2 using $\theta = \{y/w\}$:

5. $\neg happy(w)$

Resolve 5 and 1 using $\theta = \{x/w\}$:

6. $\neg graduating(w)$

Resolve 6 and 3 using $\theta = \{w/NoName1\}$:

7. $\perp$

Example 3: Proving a universally quantified sentence

- All people who are graduating are happy.
  All happy people smile.
  Everybody is graduating.

  Is everybody smiling?
- First convert to predicate logic
  $\forall x\, graduating(x) \implies happy(x)$
  $\forall x\, happy(x) \implies smiling(x)$
  $\forall x\, graduating(x)$

  $\forall x\, smiling(x)$ negate this: $\neg\forall x\, smiling(x)$
- Then convert to canonical form.

Example 3 (cont'd)

1. $\forall x \, graduating(x) \implies happy(x)$
2. $\forall x \, happy(x) \implies smiling(x)$
3. $\forall x \, graduating(x)$
4. $\neg \forall x \, smiling(x)$

Step 1. Eliminate $\implies$

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall x \, \neg happy(x) \vee smiling(x)$
3. $\forall x \, graduating(x)$
4. $\neg \forall x \, smiling(x)$

Example 3 (cont'd)

1. $\forall x\, \neg graduating(x) \vee happy(x)$
2. $\forall x\, \neg happy(x) \vee smiling(x)$
3. $\forall x\, graduating(x)$
4. $\neg \forall x\, smiling(x)$

Step 2. Move $\neg$ inwards.

1. $\forall x\, \neg graduating(x) \vee happy(x)$
2. $\forall x\, \neg happy(x) \vee smiling(x)$
3. $\forall x\, graduating(x)$
4. $\exists x\, \neg smiling(x)$

Example 3 (cont'd)

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall x \, \neg happy(x) \vee smiling(x)$
3. $\forall x \, graduating(x)$
4. $\exists x \, \neg smiling(x)$

Step 3. Standardize variables apart.

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall y \, \neg happy(y) \vee smiling(y)$
3. $\forall z \, graduating(z)$
4. $\exists w \, \neg smiling(w)$

Example 3 (cont'd)

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall y \, \neg happy(y) \vee smiling(y)$
3. $\forall z \, graduating(z)$
4. $\exists w \, \neg smiling(w)$

Step 4. Skolemize.

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall y \, \neg happy(y) \vee smiling(y)$
3. $\forall z \, graduating(z)$
4. $\neg smiling(NoName1)$

Example 3 (cont'd)

1. $\forall x \, \neg graduating(x) \vee happy(x)$
2. $\forall y \, \neg happy(y) \vee smiling(y)$
3. $\forall z \, graduating(z)$
4. $\neg smiling(NoName1)$

Step 5. Drop all $\forall$.

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(z)$
4. $\neg smiling(NoName1)$

Example 3 (cont'd)

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(z)$
4. $\neg smiling(NoName1)$

Step 6. Distribute $\wedge$ over $\vee$. (not needed)

Step 7. Make each conjuct a separate clause. (not needed)

Step 8. Standardize the variables apart again. (not needed)

Ready for resolution!

Example 3 (cont'd)

1. $\neg graduating(x) \vee happy(x)$
2. $\neg happy(y) \vee smiling(y)$
3. $graduating(z)$
4. $\neg smiling(NoName1)$

Resolve 4 and 2 using $\theta = \{y/NoName1\}$:
5. $\neg happy(NoName1)$

Resolve 5 and 1 using $\theta = \{x/NoName1\}$:
6. $\neg graduating(NoName1)$

Resolve 6 and 3 using $\theta = \{z/NoName1\}$:
7. $\perp$

More on Skolemization

If an existentially quantified variable is in the scope of universally quantified variables, then the existentially quantified variable must be a function of those other variables. We introduce a new, unique function called *Skolem function*.

For example, $\forall x\, \exists y\, (loves(x, y))$ may be replaced with any of the following:

$\forall x\, (loves(x, NoName(x)))$
$\forall x\, (loves(x, LovedOne(x)))$
$\forall x\, (loves(x, k1(x)))$

$NoName, LovedOne, k1$ are Skolem functions. They should not appear in any other sentence in the KB. They should also not have any other parameter than x.

More on conversion to CNF

Everyone who loves all animals is loved by someone.

$\forall x\, [\forall y\, Animal(y) \implies Loves(x,y)] \implies [\exists y\, Loves(y,x)]$

1. Eliminate biconditionals and implications.

   $\forall x\, [\neg \forall y\, \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\, Loves(y,x)]$

2. Move $\neg$ inwards. $(\neg \forall x\, p \equiv \exists x\, \neg p, \neg \exists x\, p \equiv \forall x \neg p)$

   $\forall x\, [\exists y\, \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\, Loves(y,x)]$

   $\forall x\, [\exists y\, \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\, Loves(y,x)]$

   $\forall x\, [\exists y\, Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\, Loves(y,x)]$

More on conversion to CNF (cont'd)

3. Standardize variables apart.
   $\forall x\, [\exists y\, Animal(y) \land \neg Loves(x, y)] \lor [\exists z\, Loves(z, x)]$
4. Skolemize.
   $\forall x\, [Animal(f(x)) \land \neg Loves(x, f(x))] \lor [Loves(g(x), x)]$
5. Drop all universal quantifiers.
   $[Animal(f(x)) \land \neg Loves(x, f(x))] \lor Loves(g(x), x)$
6. Distribute $\land$ over $\lor$.
   $[Animal(f(x)) \lor Loves(g(x), x)] \land [\neg Loves(x, f(x))$
   $\lor Loves(g(x), x)]$
7. Make each conjuct a separate clause.
   $[Animal(f(x)) \lor Loves(g(x), x)]$
   $[\neg Loves(x, f(x)) \lor Loves(g(x), x)]$

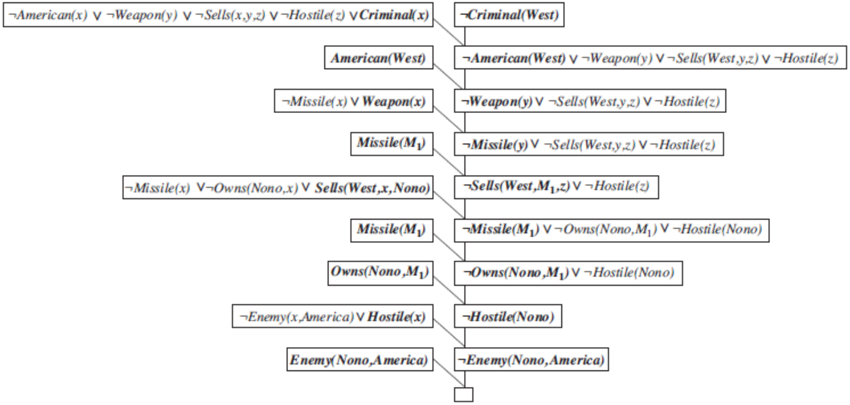
Resolution example

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
Prove that Col. West is a criminal.

Resolution example (cont'd)

- ... it is a crime for an American to sell weapons to hostile nations
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$
- Nono ... has some missiles, i.e.,
  $\exists x \; Owns(Nono, x) \wedge Missile(x)$:
  $Owns(Nono, M_1)$ and $Missile(M_1)$
- ... all of its missiles were sold to it by Colonel West
  $\forall x Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
- Missiles are weapons:
  $Missile(x) \Rightarrow Weapon(x)$
- An enemy of America counts as "hostile":
  $Enemy(x, America) \implies Hostile(x)$
- West, who is American ...
  $American(West)$
- The country Nono, an enemy of America ...
  $Enemy(Nono, America)$

Resolution example (cont'd)



The unification algorithm

function $\textsc{Unify}\ (x, y, \theta)$
returns a substitution to make x and y identical
inputs: x, a variable, constant, list, or compound expression
y, a variable, constant, list, or compound expression
θ, the substitution built up so far (optional, defaults to empty)

if $\theta =$ *failure* **then return** failure
else if $x = y$ **then return** θ
else if $\textsc{Variable?}(x)$ **then return** $\textsc{Unify-Var}(x, y, \theta)$
else if $\textsc{Variable?}(y)$ **then return** $\textsc{Unify-Var}(y, x, \theta)$
else if $\textsc{Compound?}(x)$ **and** $\textsc{Compound?}(y)$ **then**
then return $\textsc{Unify}(x.\textsc{Args},\ y.\textsc{Args},\ \textsc{Unify}(x.\textsc{Op},\ y.\textsc{Op},\ \theta))$
else if $\textsc{List?}(x)$ **and** $\textsc{List?}(y)$ **then**
then return $\textsc{Unify}(x.\textsc{Rest},\ y.\textsc{Rest},\ \textsc{Unify}(x.\textsc{First},\ y.\textsc{First},\ \theta))$
else return failure

The unification algorithm (cont'd)

function $\textsc{Unify-Var}$ (var, x, θ)
returns a substitution
    if $\{var/val\} \in \theta$ **then return** $\textsc{Unify}(val, x, \theta)$
    else if $\{x/val\} \in \theta$ **then return** $\textsc{Unify}(var, val, \theta)$
    else if $\textsc{Occur-Check?}(var, x)$ **then return** failure
    else return add $\{var/x\}$ to θ

Most general unifiers

- The unification algorithm computes a *most general unifier, mgu*, which means that it places the fewest possible restrictions on the variables
- For $p(x, y)$ and $p(joe, z)$,
  $\{x/joe, y/z\}$ and $\{x/joe, z/y\}$ are both mgu's, but
  $\{x/joe, y/joe, z/joe\}$ is not an mgu.
- **Least commitment approach**: The basic idea is to keep it as general as possible, and commit to a substitution only if you have to.

Unification algorithm examples

- $p(ruth, x)$ and $p(ruth, tia)$
  $\theta = \{x/tia\}$
  result $= p(ruth, tia)$
- $p(ruth, x)$ and $p(y, friend(y))$
  $\theta = \{y/ruth, x/friend(ruth)\}$
  result $= p(ruth, friend(ruth))$
- $parents(x, father(x), mother(bill))$ and
  $parents(bill, father(bill), y)$
  $\theta = \{x/bill, y/mother(bill)\}$
  result $= parents(bill, father(bill), mother(bill)$
- $mother(x)$ and $mother(mother(x))$
  fails due to $\textsc{Occurs-Check?}$.

Assumptions of first order logic

- Propositional calculus: no variables or functions
- Predicate calculus: allows quantified variables as parameters of predicates or functions
- Quantification can only be done over variables that represent terms
  - John likes at least one dish Jane likes.
    $\exists x \, \mathit{food}(x) \wedge \mathit{likes}(\mathit{jane}, x) \wedge \mathit{likes}(\mathit{john}, x)$
  - John likes every dish Jane likes.
    $\forall x \, (\mathit{food}(x) \wedge \mathit{likes}(\mathit{jane}, x)) \implies \mathit{likes}(\mathit{john}, x)$
  - John "does" everything Jane does
    $\forall P \, P(\mathit{jane}) \implies P(\mathit{john})$ This is <u>not</u> first order.
  - Higher order logics that allow predicates to be variables are needed to describe mathematical properties such as "every proposition implies itself" and "there are decidable propositions"

Assumptions of first order logic (cont'd)

- "Unknowns" are not explicitly represented
  unknown if not in the knowledge base
  - *Closed world assumption:* Assume true (or false) if not in the knowledge base
  - Other logics to deal with unknowns or uncertainty
    - Probability theory (degree of belief)
    - Fuzzy logic (degree of truth)
    - Dempster-Shafer theory (uncertainty and ignorance)

Summary

- Logic consists of
  - a language
    - the *syntax* tells how to build the sentences
      legal sentences are called *well-formed formulas* (wff's)
    - the *semantics* tells what the sentences mean
  - an inference procedure
    which tells us which sentences are valid inferences from other sentences

Summary (cont'd)

- The desired properties for inference procedures are:
  - sound (correct)
  - complete
  - efficient
- Universal instantiation and existential instantiation can be used to obtain a propositional KB and inferences can be made using propositional logic. This approach is slow.
- The generalized resolution inference rule provides a complete proof system for first-order logic.
- The KB has to be in conjunctive normal form (CNF). Any set of statements can be converted into CNF.
- Resolution refutation is complete, i.e., if a sentence can be entailed, it will be proven.
- Forward chaining and backward chaining are still options.

Sources for the slides

- AIMA textbook (3^{rd} edition)
- AIMA slides (http://aima.cs.berkeley.edu/)