Unit-II First Order Logic & Pi-Calculus

October 29, 2015

Herbrand Models

- For sets of clauses there are canonical interpretations called Herbrand models, which are a relatively limited set of interpretations that have the following property: If a set of clauses has a model then it has an Herbrand model
- Herbrand models will be central to the theoretical development of resolution in first-order logic

Herbrand Universes

Definition (Herbrand Universe)

Let S be a set of clauses, A the set of constant symbols in S, and F the set of function symbols in S. H_S, the Herbrand universe of S, is defined inductively:

1. $a_i \in H_S$ for $a_i \in A$,
2. $f_i^0 \in H_S$ for $f_i^0 \in F$
3. $f_i^n(t_1 , ... , t_n) \in H_S$ for $n > 1$, $f_i^n \in F$, $t_j \in H_S$
4. If there are no constant symbols or 0-ary function symbols in S, initialize the inductive definition of H S with an arbitrary constant symbol a

Note

- The Herbrand universe is just the set of ground terms that can be formed from symbols in S
- Obviously, if S contains a function symbol, the Herbrand universe is infinite since f (f (. . . (a) . . .)) $\in H_S$

Examples

1. $S_1 = \{\{p(a), \neg\, p(b), q(z)\}, \{\neg\, p(b), \neg\, q(z)\}\}$ $HS_1 = \{a, b\}$
2. $S_2 = \{\{\neg\, p(x, f(y))\}, \{p(w, g(w))\}\}$ $HS_2 = \{a, f(a), g(a), f(f(a)), g(f(a)), f(g(a)), g(g(a)), \ldots\}$
3. $S_3 = \{\{\neg\, p(a, f(x,y))\}, \{p(b, f(x,y))\}\}$ $HS_3 = \{a, b, f(a,a), f(a,b), f(b,a), f(b,b), f(a, f(a,a)), \ldots\}$

Herbrand Bases

- An alternate way of defining Herbrand models

Definition

Let H_S be the Herbrand universe for S. B_S, the Herbrand base for S, is the set of ground atomic formulas that can be formed from predicate symbols in S and terms in H_S
A relation over the Herbrand universe is simply a subset of the Herbrand base

Example

- $S_3 = \{\{\neg\ p(a,f(x,y))\}, \{p(b,f(x,y))\}\}$
- Solution:$B_{S_3} = \{p(a,\ f\ (a,\ a)),\ p(a,\ f\ (a,\ b)),\ p(a,\ f\ (b,\ a)),\ p(a,\ f\ (b,\ b)),\ .\ .\ .\ .\ ,\ p(a,\ f\ (a,\ f\ (a,\ a))),\ .\ .\ .\ .\ ,\ p(b,\ f\ (a,\ a)),\ p(b,\ f\ (a,\ b)),\ p(b,\ f\ (b,\ a)),\ p(b,\ f\ (b,\ b)),\ .\ .\ .\ .\ ,p(b,\ f\ (a,\ f\ (a,\ a))),\ .\ .\ .\}$

Assignment Due Date:29.10.2015

1. Transform the formula to clausal form:$\forall x(p(x) \rightarrow \exists y q(y))$
2. For the previous formula, describe the Herbrand universe and the Herbrand base

Resolution-First Order Logic

- The generalization of resolution to first-order logic will be done in two stages
  1. Ground Resolution
  2. Unification
- (Ground resolution rule) Let C_1, C_2 be ground clauses such that $l \in C_1$ and $l^c \in C_2$. C_1, C_2 are said to be clashing clauses and to clash on the complementary literals l, l^c . C, the resolvent of C_1 and C_2 , is the clause: $Res(C_1 , C_2) = (C_1 \ \{l\}) \cup (C_2 \ \{l^c \})$. C_1 and C_2 are the parent clauses of C

Tree representation of the ground resolution of two clauses

$$\{p(a), r(a, f(b)), r(f(a), b)\}$$

$$\{q(f(b)), r(a, f(b))\} \qquad \{p(a), \neg q(f(b)), r(f(a), b)\}$$

Substitution

Definition

A substitution of terms for variables is a set: $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$, where each x_i is a distinct variable and each t_i is a term which is not identical to the corresponding variable x_i. The empty substitution is the empty set.
Lower-case Greek letters $\{\lambda, \mu, \sigma, \theta\}$ will be used to denote substitutions. The empty substitution is denoted ϵ.

Expression, Instance

Definition

An expression is a term, a literal, a clause or a set of clauses. Let E be an expression and let $\theta = \{x_1 \leftarrow t_1 , \ldots , x_n \leftarrow t_n\}$ be a substitution. An instance $E\theta$ of E is obtained by simultaneously replacing each occurrence of x_i in E by t_i

- Example: Let $E = \{p(x), q(f(y))\}$ and a substitution $\theta = \{x \leftarrow y, y \leftarrow f(a)\}$, the instance obtained by performing the substitution is: $E\theta = \{p(y), q(f(f(a)))\}$

Composition of Substitutions

Definition

Let: $\theta = \{x_1 \leftarrow t_1 , ... , x_n \leftarrow t_n,\}$ and $\sigma = \{y_1 \leftarrow s_1, ... , y_k \leftarrow s_k \}$
be two substitutions and let $X = \{x_1 , ... , x_n \}$ and $Y = \{y_1 , ... , y_k \}$ be the sets of variables substituted for in θ and σ , respectively. $\theta\sigma$, the composition of θ and σ, is the substitution:
$\theta\sigma = \{x_i \leftarrow t_i\sigma \mid x_i \in X, x_i \neq t_i\sigma \} \cup \{y_j \leftarrow s_j \mid y_j \in Y, y_j \notin X\}$

- In otherwords, apply the substitution σ to the terms t_i of θ (provided that the resulting substitutions do not collapse to $x_i \leftarrow x_i$) and then append the substitutions from σ whose variables do not already appear in θ .

Problem

- E = p(u, v, x, y, z), θ = {x $\leftarrow$f(y), y $\leftarrow$f(a), z $\leftarrow$ u}, σ = {y $\leftarrow$ g(a), u $\leftarrow$z, v $\leftarrow$ f(f(a))} Prove that E($\theta\sigma$) = (Eθ)σ
- Refer BB

Unification

Definition

Let U $= \{A_1, \ldots, A_n\}$ be a set of atoms. A unifier θ is a substitution such that: $A_1 = \quad = A_n$

A most general unifier (mgu) for U is a unifier μ such that any unifier θ of U can be expressed as:$\theta = \mu\lambda$ for some substitution λ

- Example: Refer BB

Note

1. Not all atoms are unifiable
2. It is clearly impossible to unify atoms whose predicate symbols are different such as p(x) and q(x), as well as atoms with terms whose outer function symbols are different such as p(f (x)) and p(g(y))
3. A more tricky case is shown by the atoms p(x) and p(f (x))
4. Since x occurs within the larger term f (x), any substitution which must substitute simultaneously in both atoms cannot unify them
5. It turns out that as long as these conditions do not hold the atoms will be unifiable

General Resolution

Definition (Complement of literals)

Let $L = \{l_1, \ldots, l_n\}$ be a set of literals. Then $L^c = \{l_1^c, \ldots, l_n^c\}$

General Resolution Rule

- (General resolution rule) Let C_1, C_2 be clauses with no variables in common. Let $L_1 = \{l_1^1 , ... , l_{n_1}^1 \} \subset C_1$ and $L_2 = \{l_1^2 , ... , l_{n_2}^2\} \subset C_2$ be subsets of literals such that L_1 and L_2^c can be unified by an mgu σ . C_1 and C_2 are said to bevclashing clauses and to clash on the sets of literals L_1 and L_2. C, the resolvent of C_1 and C_2 , is the clause: $Res(C_1, C_2) = (C_1 \sigma - L_1 \sigma) \cup (C_2 \sigma - L_2 \sigma)$

General Resolution Procedure

Input : A set of clauses S
Output: If the algorithm terminates, report that the set of clauses is satisfiable or unsatisfiable

1. Let $S_0 = S$. Assume that S_i has been constructed.
2. Choose clashing clauses C_1 , $C_2 \in S_i$ and let $C = Res(C_1 , C_2)$
3. If $C = \square$, terminate and report that S is unsatisfiable
4. Otherwise, construct $S_{i+1} = S_i \cup \{C\}$
5. If $S_{i+1} = S_i$ for all possible pairs of clashing clauses, terminate and report S is satisfiable

Algorithm 1: General Resolution Procedure

Proof of Soundness

Theorem (Soundness of Resolution)

(Soundness of resolution) Let S be a set of clauses. If the empty clause $\square$ is derived when the resolution procedure is applied to S, then S is unsatisfiable.

Proof of Soundness Contd..

Proof

We need to show that if the parent clauses are (simultaneously) satisfiable, so is the resolvent.
since $\Box$ is unsatisfiable, $\implies$ that S must also be unsatisfiable.
If parent clauses are satisfiable, there is an Herbrand interpretation H such that $v_H(C_i) = T$ for i = 1, 2
The elements of the Herbrand base that satisfy C_1 and C_2 have the same form as ground atoms, so there must be a substitutions λ_i such that $C_i' = C_i\lambda_i$ are ground clauses and $v_H(C_i') = T$

Proof of Soundness Contd..

Proof.

Let C be the resolvent of C_1 and C_2.
Then there is an mgu μ for C_1 and C_2 that was used to resolve the clauses
By definition of an mgu, there must substitutions θ_i such that $\lambda_i = \sigma\theta_i$
Then $C_i' = C_i\lambda_i = C_i(\sigma\theta_i) = (C_i\sigma)\theta_i$, which shows that $C_i\sigma$ is satisfiable in the same interpretation
Let $l_1 \in C_1$ and $l_2^c \in C_2$ be the clashing literals used to derive C
Exactly one of $l_1\sigma$, $l_2^c\sigma$ is satisfiable in H
Without loss of generality, suppose that $v_H(l_1\sigma) = T$
Since $C_2\sigma$ is satisfiable, there must be a literal $l' \in C_2$ such that $l' \neq l_2^c$ and $v_H(l'\sigma) = T$
But by the construction of the resolvent, $l' \in C$ so $v_H(C) = T$. $\square$

Lifting Lemma

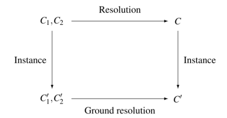
Theorem

(Lifting Lemma) Let C_1, C_2'be ground instances of C_1, C_2, respectively. Let C' be a ground resolvent of C_1' and C_2'. Then there is a resolvent C of C_1 and C_2 such that C' is a ground instance of C

Proof.

Proof

The relationships among the clauses are displayed in the following diagram



Lifting Lemma Contd..

Proof

First, standardize apart so that the names of the variables in C_1 are different from those in C_2

Let $l \in C_1'$, $l^c \in C_2'$be the clashing literals in the ground resolution.

Since C_1' is an instance of C_1 and $l \in C_1'$, there must be a set of literals $L_1 \subseteq C_1$ such that l is an instance of each literal in L_1

Similarly, there must a set $L_2 \subseteq C_2$ such that l^c is an instance of each literal in L_2

Let λ_1 and λ_2 mgus for L_1 and L_2, respectively, and let $\lambda = \lambda_1 \cup \lambda_2$. λ is a well-formed substitution since L_1 and L_2 have no variables in common

By construction, $L_1\lambda$ and $L_2\lambda$ are sets which contain a single literal each.

These literals have clashing ground instances, so they have a mgu

Lifting Lemma Contd..

Proof.

Since $L_i \subseteq C_i$, we have $L_i\lambda \subseteq C_i\lambda$.
$\therefore$ $C_1\lambda$ and $C_2\lambda$ are clauses that can be made to clash under the mgu σ.
It follows that they can be resolved to obtain clause C:
$C = ((C_1\lambda)\sigma - (L_1\lambda)\sigma\ ) \cup ((C_2\lambda)\sigma - (L_2(\lambda\sigma\ )))$
C is a resolvent of C_1 and C_2 provided that $\lambda\sigma$ is an mgu of L_1 and L_2^c
Since C_1'and C_2'are ground instances of C_1 and C_2 :
$C_1' = C_1\theta_1 = C_1\lambda\sigma\theta_1'$
$C_2' = C_2\theta_2 = C_2\lambda\sigma\theta_2'$
for some substitutions θ_1, θ_2 , θ_1', θ_2'.
Let$\theta' = \theta_1' \cup \theta_2'$. Then $C' = C\theta'$ and C' is a ground instance of C

Horne Clauses

Definition

A Horn clause is a clause of the form: $A \leftarrow B_1 , \ldots , B_n \equiv A \vee \neg B_1 , \ldots , \neg B_n$ with at most one positive literal. The positive literal A is the head and the negative literals B_i are the body

- The following terminology is used with Horn clauses:
  - A fact is a positive unit Horn clause $A \leftarrow$
  - A goal clause is a Horn clause with no positive literals $\leftarrow B_1 , \ldots , B_n$
  - A program clause is a Horn clause with one positive literal and one or more negative literals
- Program Clauses
  - $\neg (x \leq y) \vee \neg (y \leq z) \vee (x \leq z)$
- Goal Clauses
  - The formula $\neg G_1 \vee \quad \vee \neg G_n$, called a goal clause, consists entirely of negative literals, so it can only clash on the single

Note

- Logic programming prefers the use of $\leftarrow$, the reverse implication operator, to the familiar forward implication operator $\rightarrow$
- The reverse operator in $A \leftarrow B_1 , \ldots ,B_n$ has the natural reading: To prove A, prove $B_1 , \ldots , B_n$
- We can interpret this computationally as a procedure executing a sequence of statements or calling other procedures: To compute A, compute $B_1 , \ldots , B_n$

Procedure and Database

Definition

- A set of non-goal Horn clauses whose heads have the same predicate letter is a procedure.
- A set of procedures is a (logic) program.
- A procedure composed of ground facts only is a database

Undecidability of First Order Logic

- Two Register Machines:

Definition

A two-register machine M consists of two registers x and y which can store natural numbers, and a program $P = \{L_0, \dots, L_n\}$, where L_n is the in-struction halt and for $0 \leq i < n$, L_i is one of the instructions:

- $x = x+1$
- $y = y+1$
- if ($x == 0$) goto L_j ; else $x = x - 1$;
- if ($y == 0$) goto L_j ; else $y = y - 1$;

Two Register Machines Contd..

- An execution sequence of M is a sequence of states $s_k = (L_i, x, y)$, where L_i is the current instruction and x, y are the contents of the registers x and y
- s_{k+1} is obtained from s_k by executing L_i
- The initial state is $s_0 = (L_0, m, 0)$ for some m
- If for some k, $s_k = (L_n, x, y)$, the computation of M halts and M has computed $y = f(m)$

Pure First Order Logic

Definition

A formula of first-order logic is pure if it contains no function symbols (including constants which are 0-ary function symbols).