Proof Theory of FOL
Unification Algorithm

Michael Leuschel

Softwaretechnik und Programmiersrprachen

Lecture 7

09/05/2011

Resolution Principle Revisited

1. Select $p(t_1,..,t_n)$ in C_1 and $\neg p(s_1,..,s_n)$ in C_2
2. Rename variables in C_1 to fresh ones (to avoid clashes)
3. Compute a most general unifier θ of $p(t_1,..,t_n)$ and $\neg p(s_1,..,s_n)$
4. Apply θ to C_1 and C_2

5. Do resolution as before:

   cancel out $p(t_1,..,t_n)\theta$, $\neg p(s_1,..,s_n)\theta$ and keep the rest

Overview

I. Algorithm for Unification

II. Algorithm for transforming into CNF

(next lecture & tutorial)

III. Controlling Resolution: SLD

(next lecture)

Recap: Unifiers

- A substitution θ is called a **unifier** of two terms (or wff's) A and B iff $A\theta = B\theta$
- Examples

A	**B**	**a unifier θ**
p(X)	p(a)	{X/a}
p(s(0))	p(Y)	**{Y/s(0)}**
p(s(0))	p(Y)	{X/b, Y/s(0)}
p(Y,Z)	p(Z,Z)	{Y/a, Z/a}
p(Y,Z)	p(Z,Z)	**{Y/ Z}**
p(a)	p(b)	none exists (not **unifiable**)

{Y/s(0)} {X/b}
=
{X/b, Y/s(0)}

$\Rightarrow$ more general

more general

Unification: Simple Cases

- Unify variable and constant

  X a → {X/a}

- Unify two constants

  a a → {}

  a c → *fail* (not unifiable)

- Unify two variables

  X X → {}

  X Y → {X/Y}

Unification: More Complicated Cases

- Unify variable and a term with function symbols

X	s(0)	→	{X/s(0)}
X	f(a,s(0))	→	{X/f(a,s(0))}

(→ ok if no variables)

X	s(Y)	→	{X/s(Y)}
			s(Y) = s(Y) ✓
X	f(Y,s(Z))	→	{X/f(Y,s(Z))}

(→ ok if X does not occur in the term)

X	s(X)	→	{X/s(X)} ??

Unification: Occur Check

- Is the following a valid unifier:

  X s(X) → {X/s(X)} ??

- **NO**:

  X{X/s(X)} = **s(X)**

  s(X){X/s(X)} = **s(s(X))**

  Different !

- There exists NO unifier !
- Same holds for **any** term t which contains X

- → Called the **Occur Check**

Unification: Function Symbols

- Unify two terms with function symbols

f(a)	**g**(a)	→	*fail*
f(a)	**c**	→	*fail*
c	**f**(X)	→	*fail*

(→ fail if **different** top-level function symbol)

f(a)	f(a)	→	{}
f(a)	f(X)	→	{X/a}
f(a,X)	f(a,s(0))	→	{X/s(0)}

(→ unify arguments if **same** function symbol)

Exercise

- Try to unify the following terms

f(f(X))	f(g(X))	→
f(s(X), s(Z))	f(Y,s(0))	→
f(X,X)	f(Y,s(0))	→

Solution

- Try to unify the following terms

f(f(X))	f(g(X))	→ *fail*
f(s(X), s(Z))	f(Y,s(0))	→ {Y/s(X), Z/0}
f(s(X), s(0))		
f(X,X)	f(Y,s(0))	→ {X/s(0), Y/s(0)}
f(s(0), s(0))		not {X/Y, Y/s(0)} !

Disagreement Tuple

- Given two terms (or wff's) A and B
- Locate the **leftmost** symbol position at which A and B do **not** have the **same** symbol
- Extract the **subexpressions** A',B' in A and B starting at that position
- ⟨A',B'⟩ is the **disagreement tuple**
- Examples:
  - f(Y,s(0)) f(a,f(a)) → ⟨Y, a⟩
  - f(Y,s(Y)) f(Y,s(s(0))) → ⟨Y, s(0)⟩
  - f(Y,s(Y)) f(X,s(s(0))) → ⟨Y, X⟩

The Unification Algorithm

1. Put k=0 and θ_0 = {}
2. **If** $A\theta_k = B\theta_k$ **then** stop; θ_k is an **mgu** of A and B.
   Else find the disagreement tuple $\langle a_k, b_k \rangle$ of $A\theta_k, B\theta_k$
3. **If** a_k is a variable which does not occur in b_k then put $\theta_{k+1} = \theta_k\{a_k / b_k\}$, increment k and go to 2.
   Else if b_k is a variable which does not occur in a_k then put $\theta_{k+1} = \theta_k\{b_k / a_k\}$, increment k and go to 2.
   Otherwise, stop; A and B are not **unifiable**

Exercise

- Trace the algorithm for unifying
  f(Y,s(Y))   f(X,s(s(0)))

Trace of the Algorithm

- $A = f(Y, s(Y))$ $B = f(X, s(s(0)))$ $\theta_0 = \{\}$
- Iteration 1:
  - $A\theta_0 = f(\underline{Y}, s(Y))$ $B\theta_0 = f(\underline{X}, s(s(0)))$
  - $\langle A_0, B_0 \rangle = \langle Y, X \rangle$ $\rightarrow$ $\boldsymbol{\theta_1 = \{Y/X\}}$
- Iteration 2:
  - $A\theta_1 = f(X, s(\underline{X}))$, $B\theta_1 = f(X, s(\underline{s(0)}))$
  - $\langle A_1, B_1 \rangle = \langle X, s(0) \rangle$
  - $\rightarrow$ $\boldsymbol{\theta_2 =} \{Y/X\}\{X/s(0)\} = \boldsymbol{\{Y/s(0), X/s(0)\}}$
- We have found an mgu θ_2 :
  - $A\theta_2 = f(s(0), s(s(0)))$, $B\theta_2 = f(s(0), s(s(0)))$

Properties of the Algorithm

- Robinson, 1963 *(J. ACM, 1965)*

- If the terms (or wff's) are unifiable, then it produces a **most general unifier** (**mgu**)
  If the terms (or wff's) are not unifiable, then it reports failure.

- Furthermore, this mgu is **unique** up to variable renaming (Unification Theorem)

Other Unification Algorithms

- Paterson/Wegman 1978 *(J. Comp. Sys. Sciences)*
- Martelli/Montanari 1982 *(ACM Toplas)*
  - Both linear
- Prolog
  - Basically use Robinson's algorithm
  - occur check is expensive (worst case: exponential)
  - ⇒ Not performed by Prolog systems !!
    - `X = f(X)` ➔ yes
    - **Cyclic** Structures !
    - `print(X)` ➔ f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f …

Summary

- Unification Algorithm
  - Disagreement Tuple
  - Occur Check
  - Robinson's Algorithm

- Still to be seen:
  - How to transform FOL into CNF
  - How Prolog controls resolution:
    which literals to be selected for resolution/
    unification