Unification Algorithm

ChuChen

Definition

The unification problem in first-order logic can be expressed as follows: Given two terms containing some variables, find, if it exists, the *simplest substitution* (i.e., an assignment of some term to every variable) which makes the two terms equal. The resulting substitution is called the *most general unifier* .

Example

$f(x_1, h(x_1), x_2\} = f(g(x_3), x_4, x_3)$

$б_1 = ((g(x_3), x_1), (x_3, x_2), (h(g(x_3), x_4)\}$

$б_2 = ((g(a), x_1), (a, x_2), (a, x_3), (h(g(a)), x_4)\}.$

Definition

$A = \bigcup A_i \quad (A_i \cap A_j = \emptyset, i \neq j) \quad i=0,1 \ldots n,$ be a ranked alphabet, where A_i contains the i-adic function symbols (the elements of A_0 are constant symbols).

Furthermore, let V be the alphabet of the variables.

Terms

The terms are defined recursively as follows:

(1) constant symbols and variables are terms;

(2) if $t_1 \ldots\ldots t_n$ $(n \geq 1)$ are terms and $f \in A_n$, then $f(t_1, \ldots, t_n)$ is a term.

Substitution

- A substitution б is a mapping from variables to terms,
- A substitution can be represented by a finite set of ordered pairs б= {(t_1, x_1), (t_2, x_2) , . . . , (t_m, X_m)} where t_i are terms and x_i are distinct variables, i = 1 , . . . , m.

Substitution

- To apply a substitution б to a term t, we simultaneously substitute all occurrences in t of every variable x_i in a pair (t_i, x_i) of б with the corresponding term t_i. We call the resulting term $t_б$.
- For instance, given a term $t = f(x_1, g(x_2, a))$ *and a substitution* $б = \{(h(x_2), x_1), (b, x_2)\}$, we have $t_б = f(h(x_2), g(b), a)$ *and* $t_{бб} = f(h(b), g(b), a)$.

Unifier

- The standard unification problem can be written as an equation

$$t' = t''.$$

A solution of the equation, called a unifier, is any substitution б, if it exists, which makes the two terms identical.

For instance, two unifiers of the equation

$f(x_1, h(x_1), x_2\} = f(g(x_3), x_4, x_3)$

are $б_1 = ((g(x_3), x_1), (x_3, x_2), (h(g(x_3), x_4)\}$ and $б_2 = ((g(a), x_1), (a, x_2), (a, x_3), (h(g(a)), x_4)\}$.

Sets of equations

- In what follows it is convenient also to consider sets of equations:
- $t_1' = t_1''$,

  .....

  $t_j' = t_j''$,

  .....

  $t_n' = t_n''$,

  $j = 1 \ldots . . . n.$

Transformations of Sets of Multi-equations

- (1)Term Reduction.

  Let

  $f(t_1', t_2', \ldots, t_n',) = f(t_1'', t_2'' \ldots\ldots t_n'')$, $f \in A_n$ be an equation where both terms are not variables and where the two root function symbols are equal. The new set of equations is obtained by replacing this equation with the following ones:

  $t_1' = t_1''$

  $t_2' = t_2''$

  .....

  .....

  $t_n' = t_n''$

- If n = 0, then f is a constant symbol, and the equation is simply erased.

Transformations of Sets of Multi-equations

- (2) *Variable Elimination.*

  Let x = t be an equation where x is a variable and t is any term (variable or not). The new set of equations is obtained by applying the substitution б= ((t, x)} to both terms of all other equations in the set (without erasing x = t).

The MGU

- A set of equations is said to be in solved form iff it satisfies the following conditions:

  (1) the equations are $x_j = t_j$, j = 1, . . . , k;

  (2) every variable which is the left member of some equation occurs only there.

  A set of equations in solved form has the obvious unifier

  $\ell = \{(t_1, x_1), (t_2, x_2), \ldots\ldots , (t_k, x_k)\}$.

  If there is any other unifier, it can be obtained as

  $\ell = \{ (t_1, x_1), (t_2, x_2) . . . . , (t_k, x_k)\}$ U a

  where a is any substitution which does not rewrite variables xl . . . . , xk. Thus ℓ is called a *most general unifier* (*mgu*).

Algorithm 1

- (a) Select any equation of the form

  t=x where t is not a variable and x is a variable, and rewrite it as x=t
- (b) Select any equation of the form
- x=x
- where x is variable, and erase it.
- (c) Select any equation of the form t' = t"

Algorithm 1

where t' and *t" are not variables.*

If

the two root function symbols are different,

**for example f(x , y) ,g(x , y)*

stop with failure;

else

apply term reduction. //iteration

Algorithm 1

- (d) Select any equation of the form

$$x=t$$

where x is a variable which occurs somewhere else in the set of equations and where $t \neq x$. If x occurs in t, then stop with failure; otherwise, apply variable elimination.

Related Definition

- A multiset is a family of elements in which no ordering exists but in which many identical elements may occur.
- A *multi-equation* is the generalization of an equation, and it allows us to group together many terms which should be unified. To represent multi-equations we use the notation S = M where the left-hand side S is a nonempty set of variables and the right-hand side M is a multi-set 1 of nonvariable terms.

Example

- An example is $\{x_1, x_2, x_3\} = (t_1, t_2)$.

  $x_1 = x_2$;

  $x_3 = x_1$;

  $t_1 = x_1$;

  $x_2 = t_2$;

  $t_1 = t_2$;

Common Part

- *The common part* of a multiset of terms M is a *term* which intuitively, is obtained by superimposing all terms of M and by taking the part which is common to all of them starting from the root.
- For instance, given the multiset of terms ($f(x_1, g(a, f(x_5, b)))$, $f(h(c), g(x_2, f(b, x_5)))$, $f(h(x_4), g(x_6, x_3))$), the common part is $f(x_1, g(x2, x3))$.
- See PDF Page 8.

Frontier

- *The frontier* is a set of *multi-equations*, where every multi-equation is associated with a leaf of the common part and consists of all sub terms (one for each term of M) corresponding to that leaf.

Transformations of Sets of Multi-Equations

- *Multi-equation Reduction:*

  Let Z be a set of multi-equations containing a multi-equation S = M such that M is nonempty and has a common part C and a frontier F. The new set *Z' of multi-equations is obtained by replacing S = M with* the union of the multi-equation S = (C) and of all the multi-equations of F:

$$Z=(Z- \{S=M\})U\{S=(C)\} U F .$$

Transformations of Sets of Multi-Equations

- We say that a set Z of multi-equations is compact iff $\forall (S = M), (S' = M') \in Z: S \cap S' = \emptyset$.
- Compactification :

  Let Z be a noncompact set of multi-equations.

  Let R be a relation between pairs of multi-equations of Z such that (S = M) R (S' = M') iff $S \cap S' \neq \emptyset$, The relation R partitions the set Z into equivalence classes.

Transformations of Sets of Multi-Equations

- To obtain the final compact set Z', all multi-equations belonging to the same equivalence class are transformed into single multiequations by taking the union of their left-hand and right-hand sides.

Structure

- *A* system R is a pair (T, U), where T is a sequence and U is a set of multi-equations

  (either possibly empty):

  T part, that is, the *triangular or solved part of R.*

  U part, that is the *unsolved part.*

Example

- U: ={

  {x} = $(f(x_1, g(x_2, x_3), x_2, b), f(g(h(a, x_5), x_2), x_1, h(a, x4), x4))$,

  $\{x_1\} = \emptyset$,

  $\{x_2\} = \emptyset$,

  $\{x_3\} = \emptyset$,

  $\{x_4\} = \emptyset$,

  $\{x_5\} = \emptyset$

  };
- T:().

Algorithm 2

- *Algorithm 2*
- (1) repeat:
- (1.1) Select a multi-equation S = M of U with M ≠Ø.
- (1.2) Compute the common part C and the frontier F of M. If M has no common part, stop with failure (clash).
- (1.3) If the left-hand sides of the frontier of M contain some variable of S, stop with failure (cycle).
- (1.4) Transform U using multi-equation reduction on the selected multi-equation and compactification.
- (1.5) Let S = {X_1 . . . . . X_n). *Apply the substitution* $\sigma = \{(C, x_1) . . . . . (C, x_n)\}$ *to all* terms in the right-hand side of the multi-equations of U.
- (1.6) Transfer the multi-equation S = (C) from U to the end of T.
  until the U part of R contains only multi-equations, if any, with empty right-hand sides.
- (2) Transfer all the multi-equations of U (all with M = Ø) to the end of T, and stop with success.

Example

- U: ={

  {x} = $(f(x_1, g(x_2, x_3), x_2, b), f(g(h(a, x_5), x_2), x_1, h(a, x_4), x_4)),$
- $\{x_1\} = \emptyset,$
- $(x_2\} = \emptyset,$
- $\{x_3\} = \emptyset,$
- $(x_4\} = \emptyset,$
- $\{x_5\} = \emptyset$
- };
- T:().

Example

- After the first iteration of Algorithm 2 we get
- U: ={
- {x} = $(g(h(a, x_5), x_2), g(x_2, x_3))$,
- $\{x_2\} = \{h(a, x_4)\}$,
- $\{x_3\} = \emptyset$,
- $\{x_4\} = (b)$,
- $\{x_5\} = \emptyset$
- };
- T: ({x} = $(f(x_1, x_1, x_2, x_4))$.

Example

- We now eliminate variable x2, obtaining
- U: ={
- $\{x_1\} = (g(h(a, x_5), h(a, x_4)), g(h(a, x_4), x_3))$,
- $\{x_3\} = \emptyset$,
- $\{x_4\} = (b)$,
- $\{x_5\} = \emptyset\}$;
- T: =($\{x\} = (f (x_1, x_1, x_2, x_4))$,
- $\{x_2\} = (h(a, x_4)))$.

Improvements

- We now define a relation on the classes *Si of this partition: we say that* $S_i < S_j$ *iff there exists a variable* of S_i *occurring in some term of* M_j*, where* M_j *is the right-hand side of* the multi-equation whose left-hand side is S_j.

Improvements

- THEOREM:

  If a system R has a unifier, then the relation < is a partial ordering.*

- PROOF:

  If $S_i < S_j$, *then, in all unifiers of the system, the term substituted for* every variable in *Si must be a strict subterm of the term substituted for every* variable in S_j. *Thus, if the system has a unifier, the graph of the relation < can not* have cycles. Therefore, its transitive closure must be a partial ordering.

Improvements

- COROLLARY:

  If the system R has a unifier and its U part is nonempty, there exists a multi-equation S = M such that the variables in S do not occur elsewhere in U.

- PROOF: Let S = M be a multi-equation such that S is "on top" of the partial ordering < (i.e., $\sim\exists S_i$, $S < S_i$). *The variables in S occur neither in the other left-hand* sides of U (since they are disjoint) nor in any right member M_i of U, since otherwise S < *Si.*

Algorithm 3

- (1) repeat
- (1.1) Select a multi-equation S = M of U such that the variables in S do not occur
- elsewhere in U. If a multi-equation with this property does not exist, stop with
- failure (cycle).
- (1.2) if M is empty
- then
- transfer this multi-equation from U to the end of T.
- else begin
- (1.2.1) Compute the common part C and the frontier F of M. If M has
- no common part, stop with failure (clash).
- (1.2.2) Transform U using multi-equation reduction on the selected
- Multi-equation and compactification.
- (1.2.3) Transfer the multi-equation S = (C) from U to the end of T.
- end
- until the U part of R
- (2) stop with success.

The Counter

- In this section we show how to implement efficiently the operation of selecting a multi-equation "on top" of the partial ordering in step (1.1) of Algorithm 3.
- The idea is to associate with every multi-equation *a counter which contains the* number of other occurrences in U of the variables in its left-hand side.

The Counter

- This counter is initialized by scanning the whole U part at the beginning.

- A multi-equation whose counter is set to zero is on top of the partial ordering and it is first to be selected.

Improvements

- U: =
- {
- [0] {x} =(f(x_1 ,g(x_2, x_3), x_2, b), f(g(h(a, x_5), x_2),x_1 ,h(a, x4), x4)),
- [2] {x_1} = Ø,
- [3] {x_2} = Ø,
- [1] (x_3} = Ø,
- [2] {x_4} = Ø,
- [1] {x_5} = Ø
- };
- T:( ).

Improvements

- The next steps are as follows:
- U:= {
- [0] $\{x_1\} = (g(h(a, x_5), x_2), g(x_2, x_3))$,
- [2] $\{x_2\} = (h(a, x_4))$,
- [1] $(x_3) = \emptyset$,
- [1] $\{x_4\} = (b)$,
- [1] $\{x_5\} = \emptyset\}$;
- T: { $\{x\} = (f(x_1, x_1, x_2, x_4))\}$.

Improvements

- U: ={
- [0] $\{x_2, x_3\} = (h(a, x_4), h(a, x_5))$,
- [1] $\{x_4\} = (b)$,
- [1] $\{x_5\} = \emptyset$
- };
- T:= $(\{x\} = (f(x_1, x_1, x_2, x_4))$,
- $\{x_1\} = (g(x_2, x_3)))$.

Consistency

- In the case of nonunifying data, Algorithm 3 can stop with failure in two ways:
- (1) in step (1.1) if a cycle has been detected,
  (2) in step (1.2.1) if a clash occurs.
- The latter kind of failure can be detected without altering the structure of the algorithm.

Consistency Checker

- Let us first give the following definition. Two terms are *consistent :*

  iff either at least one of them is a variable or they are both none variable terms with the same root function symbol and pair wise consistent arguments.

Consistency Checker

- *A multi-term can be either empty or of the form* $f(P_1 . . . . . P_n)$ *where* $f \in A_n$ *, and* P_i $(i = 1 . . . . . n)$ *is a pair* (S_i, M_i) *consisting of a set of variables* S_i *and a* multi-term M_i*. Furthermore,* S_i *and* M_i *can not both be empty.*
- For instance, the multi-set of consistent terms *(f(x, g(a, y)), f(b, x), f(x, y))* can be represented with the multi-term :

  f(({x}, b), *({x , y},g((Ø, a), ({y}, Ø))))*.