COSE212: Programming Languages

Lecture 13 — Automatic Type Inference (1)

Hakjoo Oh
2020 Fall

The Problem of Automatic Type Inference

Given a program E, infer the most general type of E if E can be typed (i.e., $[] \vdash E : t$ for some $t \in T$). If E cannot be typed, say so.

- $\texttt{let}\ f = \texttt{proc}\ (x)\ (x + 1)\ \texttt{in}\ (\texttt{proc}\ (x)\ (x\ 1))\ f$
- $\texttt{let}\ f = \texttt{proc}\ (x)\ (x + 1)\ \texttt{in}\ (\texttt{proc}\ (x)\ (x\ \mathit{true}))\ f$
- $\texttt{proc}\ (x)\ x$

Automatic Type Inference

- A static analysis algorithm that automatically figures out types of expressions by observing how they are used.
- The algorithm is *sound and complete* with respect to the type system design.
  - (Sound) If the analysis finds a type for an expression, the expression is well-typed with the type according to the type system.
  - (Complete) If an expression has a type according to the type system, the analysis is guaranteed to find the type.
- The algorithm consists of two steps:
  1. Generate type equations from the program text.
  2. Solve the equations.

Generating Type Equations

For every subexpression and variable, introduce type variables and derive equations between the type variables.

Example 1

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ \underbrace{\texttt{proc}\ (\underbrace{x}_{t_x})\ \underbrace{(\underbrace{(f\ 3)}_{t_3} - \underbrace{(f\ x)}_{t_4})}_{t_2}}_{t_1}}_{t_0}$$

$$\begin{array}{rcl}
t_0 & = & t_f \to t_1 \\
t_1 & = & t_x \to t_4 \\
t_3 & = & \mathsf{int} \\
t_4 & = & \mathsf{int} \\
t_2 & = & \mathsf{int} \\
t_f & = & \mathsf{int} \to t_3 \\
t_f & = & t_x \to t_4
\end{array}$$

Example 2

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ (\underbrace{f\ 11}_{t_1})}_{t_0}$$

$$\begin{array}{rcl} t_0 & = & t_f \rightarrow t_1 \\ t_f & = & \mathsf{int} \rightarrow t_1 \end{array}$$

Example 3

$$\underbrace{\texttt{if}\ \underbrace{x}_{t_x}\ \texttt{then}\ \underbrace{(x-1)}_{t_1}\ \texttt{else}\ 0}_{t_0}$$

$$\begin{array}{rcl} t_x & = & \text{bool} \\ t_1 & = & t_0 \\ \text{int} & = & t_0 \\ t_x & = & \text{int} \\ t_1 & = & \text{int} \end{array}$$

Example 4

$$\underbrace{\texttt{proc}\ (\underbrace{\ f\ }_{t_f})\ \underbrace{(\texttt{iszero}\ \underbrace{(f\ f)}_{t_2})}_{t_1}}_{t_0}$$

$$\begin{array}{rcl}
t_0 & = & t_f \rightarrow t_1 \\
t_1 & = & \mathsf{bool} \\
t_2 & = & \mathsf{int} \\
t_f & = & t_f \rightarrow t_2
\end{array}$$

Idea: Deriving Equations from Typing Rules

For each expression e and variable x, let t_e and t_x denote the type of the expression and variable. Then, the typing rules dictate the equations that must hold between the type variables.

- $$\frac{\Gamma \vdash E_1 : \mathsf{int} \qquad \Gamma \vdash E_2 : \mathsf{int}}{\Gamma \vdash E_1 + E_2 : \mathsf{int}}$$

  $$t_{E_1} = \mathsf{int} \ \wedge \ t_{E_2} = \mathsf{int} \ \wedge \ t_{E_1+E_2} = \mathsf{int}$$

- $$\frac{\Gamma \vdash E : \mathsf{int}}{\Gamma \vdash \mathtt{iszero}\ E : \mathsf{bool}}$$

  $$t_E = \mathsf{int} \ \wedge \ t_{(\mathtt{iszero}\ E)} = \mathsf{bool}$$

- $$\frac{\Gamma \vdash E_1 : t_1 \to t_2 \qquad \Gamma \vdash E_2 : t_1}{\Gamma \vdash E_1\ E_2 : t_2}$$

  $$t_{E_1} = t_{E_2} \to \ t_{(E_1\ E_2)}$$

Idea: Deriving Equations from Typing Rules

- $$\frac{\Gamma \vdash E_1 : \text{bool} \qquad \Gamma \vdash E_2 : t \qquad \Gamma \vdash E_3 : t}{\Gamma \vdash \texttt{if } E_1 \texttt{ then } E_2 \texttt{ else } E_3 : t}$$

$$\begin{array}{rcl} t_{E_1} & = & \text{bool} \wedge \\ t_{E_2} & = & t_{(\texttt{if } E_1 \texttt{ then } E_2 \texttt{ else } E_3)} \wedge \\ t_{E_3} & = & t_{(\texttt{if } E_1 \texttt{ then } E_2 \texttt{ else } E_3)} \end{array}$$

- $$\frac{[x \mapsto t_1]\Gamma \vdash E : t_2}{\Gamma \vdash \texttt{proc } x\ E : t_1 \rightarrow t_2}$$

$$t_{(\texttt{proc } (x)\ E)} = t_x \rightarrow t_E$$

- $$\frac{\Gamma \vdash E_1 : t_1 \qquad [x \mapsto t_1]\Gamma \vdash E_2 : t_2}{\Gamma \vdash \texttt{let } x = E_1 \texttt{ in } E_2 : t_2}$$

$$t_x = t_{E_1} \ \wedge \ t_{E_2} = t_{(\texttt{let } x = E_1 \texttt{ in } E_2)}$$

Summary

The algorithm for automatic type inference:

1. Generate type equations from the program text.
    - Introduce type variables for each subexpression and variable.
    - Generate equations between type variables according to typing rules.
2. Solve the equations.

COSE212: Programming Languages

Lecture 14 — Automatic Type Inference (2)

Hakjoo Oh
2020 Fall

Goal

- So far we have informally discussed how to derive type equations.
- In this lecture, we define the procedure precisely.

Language

$$
\begin{array}{rcl}
E & \to & n \\
  & | & x \\
  & | & E + E \\
  & | & E - E \\
  & | & \texttt{iszero}\ E \\
  & | & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E \\
  & | & \texttt{let}\ x = E\ \texttt{in}\ E \\
  & | & \texttt{proc}\ x\ E \\
  & | & E\ E
\end{array}
$$

$$
\begin{array}{rcl}
T & \to & \mathsf{int} \\
  & | & \mathsf{bool} \\
  & | & T \to T \\
  & | & \alpha\ (\in TyVar)
\end{array}
$$

Type Equations

- Type equations are conjunctions of "type equalities": e.g.,

$$\begin{array}{rcl} t_0 & = & t_f \to t_1 \\ t_1 & = & t_x \to t_4 \\ t_3 & = & \mathsf{int} \\ t_4 & = & \mathsf{int} \\ t_2 & = & \mathsf{int} \\ t_f & = & \mathsf{int} \to t_3 \\ t_f & = & t_x \to t_4 \end{array}$$

- Type equations ($TyEqn$) are defined inductively:

$$\begin{array}{rcl} TyEqn & \to & \emptyset \\ & | & T \doteq T \ \wedge \ TyEqn \end{array}$$

Deriving Type Equations

- Algorithm for generating equations:

$$\mathcal{V} : (\mathit{Var} \to T) \times E \times T \to \mathit{TyEqn}$$

- $\mathcal{V}(\Gamma, e, t)$ generates the condition for e to have type t in Γ:

$$\Gamma \vdash e : t \text{ iff } \mathcal{V}(\Gamma, e, t) \text{ is satisfied.}$$

- Examples:
  - $\mathcal{V}([x \mapsto \text{int}], \texttt{x+1}, \alpha) =$
  - $\mathcal{V}(\emptyset, \texttt{proc}\ (x)\ (\texttt{if}\ x\ \texttt{then}\ 1\ \texttt{else}\ 2), \alpha \to \beta) =$
- To derive type equations for closed expression E, we call $\mathcal{V}(\emptyset, E, \alpha)$, where α is a fresh type variable.

Deriving Type Equations

$$
\begin{aligned}
\mathcal{V}(\Gamma, n, t) &= \\
\mathcal{V}(\Gamma, x, t) &= \\
\mathcal{V}(\Gamma, e_1 + e_2, t) &= \\
\mathcal{V}(\Gamma, \texttt{iszero}\ e, t) &= \\
\mathcal{V}(\Gamma, \texttt{if}\ e_1\ e_2\ e_3, t) &= \\
\mathcal{V}(\Gamma, \texttt{let}\ x = e_1\ \texttt{in}\ e_2, t) &= \\
\mathcal{V}(\Gamma, \texttt{proc}\ (x)\ e, t) &= \\
\\
\mathcal{V}(\Gamma, e_1\ e_2, t) &=
\end{aligned}
$$

Example

$$
\begin{array}{ll}
\mathcal{V}(\emptyset, (\texttt{proc } (x)\ (x))\ 1, \alpha) & \\
= \mathcal{V}(\emptyset, \texttt{proc } (x)\ (x), \alpha_1 \to \alpha) \land \mathcal{V}(\emptyset, 1, \alpha_1) & \text{new } \alpha_1 \\
= \alpha_1 \to \alpha \doteq \alpha_2 \to \alpha_3 \land \mathcal{V}([x \mapsto \alpha_2], x, \alpha_3) \land \alpha_1 \doteq \text{int} & \text{new } \alpha_2, \alpha_3 \\
= \alpha_1 \to \alpha \doteq \alpha_2 \to \alpha_3 \land \alpha_2 \doteq \alpha_3 \land \alpha_1 \doteq \text{int} &
\end{array}
$$

Exercise 1

$$\mathcal{V}(\emptyset, \texttt{proc}\ (f)\ (f\ 11), \alpha)$$

Exercise 2

$$\mathcal{V}([x \mapsto \mathsf{bool}], \texttt{if } x \texttt{ then } (x - 1) \texttt{ else } \mathbf{0}, \alpha)$$

Exercise 3

$$\mathcal{V}(\emptyset, \texttt{proc}\ (f)\ (\texttt{iszero}\ (f\ f)), \alpha)$$

Summary

We have defined the algorithm for deriving type equations from program text:

- Given a program E, call $\mathcal{V}(\emptyset, E, \alpha)$ to derive type equations.
- Solve the equations and find the type assigned to α.

COSE212: Programming Languages

Lecture 15 — Automatic Type Inference (3)

Hakjoo Oh
2020 Fall

Finding a Solution of Type Equations

Find the values of type variables that make all the equations true.

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ \underbrace{\texttt{proc}\ (\underbrace{x}_{t_x})\ \underbrace{(\underbrace{(f\ 3)}_{t_3} - \underbrace{(f\ x)}_{t_4})}_{t_2}}_{t_1}}_{t_0}$$

Equations	Solution
$t_0 = t_f \to t_1$	$t_0 = (\text{int} \to \text{int}) \to (\text{int} \to \text{int})$
$t_1 = t_x \to t_2$	$t_1 = \text{int} \to \text{int}$
$t_3 = \text{int}$	$t_2 = \text{int}$
$t_4 = \text{int}$	$t_3 = \text{int}$
$t_2 = \text{int}$	$t_4 = \text{int}$
$t_f = \text{int} \to t_3$	$t_f = \text{int} \to \text{int}$
$t_f = t_x \to t_4$	$t_x = \text{int}$

Static type systems find such a solution using *unification algorithm*.

Example 1

The calculation is split into equations to be solved and substitution found so far. Initially, the substitution is empty:

Equations	Substitution
$t_0 = t_f \rightarrow t_1$	
$t_1 = t_x \rightarrow t_2$	
$t_3 = \mathsf{int}$	
$t_4 = \mathsf{int}$	
$t_2 = \mathsf{int}$	
$t_f = \mathsf{int} \rightarrow t_3$	
$t_f = t_x \rightarrow t_4$	

Example 1

Consider each equation in turn. If the equation's left-hand side is a variable, move it to the substitution:

Equations	Substitution
$t_1 = t_x \rightarrow t_2$	$t_0 = t_f \rightarrow t_1$
$t_3 = \text{int}$	
$t_4 = \text{int}$	
$t_2 = \text{int}$	
$t_f = \text{int} \rightarrow t_3$	
$t_f = t_x \rightarrow t_4$	

Example 1

Move the next equation to the substitution and propagate the information to the existing substitution (i.e., substitute the right-hand side for each occurrence of t_1):

Equations	Substitution
$t_3 = \text{int}$	$t_0 = t_f \rightarrow (t_x \rightarrow t_2)$
$t_4 = \text{int}$	$t_1 = t_x \rightarrow t_2$
$t_2 = \text{int}$	
$t_f = \text{int} \rightarrow t_3$	
$t_f = t_x \rightarrow t_4$	

Example 1

Same for the next three equations:

Equations	Substitution
$t_4 = \mathsf{int}$	$t_0 = t_f \to (t_x \to t_2)$
$t_2 = \mathsf{int}$	$t_1 = t_x \to t_2$
$t_f = \mathsf{int} \to t_3$	$t_3 = \mathsf{int}$
$t_f = t_x \to t_4$	

Equations	Substitution
$t_2 = \mathsf{int}$	$t_0 = t_f \to (t_x \to t_2)$
$t_f = \mathsf{int} \to t_3$	$t_1 = t_x \to t_2$
$t_f = t_x \to t_4$	$t_3 = \mathsf{int}$
	$t_4 = \mathsf{int}$

Equations	Substitution
$t_f = \mathsf{int} \to t_3$	$t_0 = t_f \to (t_x \to \mathsf{int})$
$t_f = t_x \to t_4$	$t_1 = t_x \to \mathsf{int}$
	$t_3 = \mathsf{int}$
	$t_4 = \mathsf{int}$
	$t_2 = \mathsf{int}$

Example 1

Consider the next equation $t_f = \text{int} \rightarrow t_3$. The equation contains t_3, which is already bound to int in the substitution. Substitute int for t_3 in the equation. This is called *applying* the substitution to the equation.

Equations	Substitution
$t_f = \text{int} \rightarrow \text{int}$	$t_0 = t_f \rightarrow (t_x \rightarrow \text{int})$
$t_f = t_x \rightarrow t_4$	$t_1 = t_x \rightarrow \text{int}$
	$t_3 = \text{int}$
	$t_4 = \text{int}$
	$t_2 = \text{int}$

Move the resulting equation to the substitution and update it.

Equations	Substitution
$t_f = t_x \rightarrow t_4$	$t_0 = (\text{int} \rightarrow \text{int}) \rightarrow (t_x \rightarrow \text{int})$
	$t_1 = t_x \rightarrow \text{int}$
	$t_3 = \text{int}$
	$t_4 = \text{int}$
	$t_2 = \text{int}$
	$t_f = \text{int} \rightarrow \text{int}$

Example 1

Apply the substitution to the equation:

Equations	Substitution
$\text{int} \to \text{int} = t_x \to \text{int}$	$t_0 = (\text{int} \to \text{int}) \to (t_x \to \text{int})$
	$t_1 = t_x \to \text{int}$
	$t_3 = \text{int}$
	$t_4 = \text{int}$
	$t_2 = \text{int}$
	$t_f = \text{int} \to \text{int}$

If neither side of the equation is a variable, simplify the equation by yielding two new equations:

Equations	Substitution
$\text{int} = t_x$	$t_0 = (\text{int} \to \text{int}) \to (t_x \to \text{int})$
$\text{int} = \text{int}$	$t_1 = t_x \to \text{int}$
	$t_3 = \text{int}$
	$t_4 = \text{int}$
	$t_2 = \text{int}$
	$t_f = \text{int} \to \text{int}$

Example 1

Switch the sides of the first equation and move it to the substitution:

Equations	Substitution
$\text{int} = \text{int}$	$t_0 = (\text{int} \rightarrow \text{int}) \rightarrow (\text{int} \rightarrow \text{int})$
	$t_1 = \text{int} \rightarrow \text{int}$
	$t_3 = \text{int}$
	$t_4 = \text{int}$
	$t_2 = \text{int}$
	$t_f = \text{int} \rightarrow \text{int}$
	$t_x = \text{int}$

The final substitution is the solution of the original equations.

Example 2

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ (\underbrace{f\ 11}_{t_1})}_{t_0}$$

$$\begin{array}{rcl} t_0 & = & t_f \rightarrow t_1 \\ t_f & = & \text{int} \rightarrow t_1 \end{array}$$

Example 2

1.

Equations	Substitution
$t_0 = t_f \rightarrow t_1$	
$t_f = \text{int} \rightarrow t_1$	

2.

Equations	Substitution
$t_f = \text{int} \rightarrow t_1$	$t_0 = t_f \rightarrow t_1$

3.

Equations	Substitution
	$t_0 = (\text{int} \rightarrow t_1) \rightarrow t_1$
	$t_f = \text{int} \rightarrow t_1$

The type is *polymorphic* in t_1.

Example 3

$$\underbrace{\texttt{if}\ \underbrace{x}_{t_x}\ \texttt{then}\ \underbrace{(x-1)}_{t_1}\ \texttt{else}\ 0}_{t_0}$$

$$\begin{array}{rcl} t_x & = & \mathsf{bool} \\ t_1 & = & t_0 \\ \mathsf{int} & = & t_0 \\ t_x & = & \mathsf{int} \\ t_1 & = & \mathsf{int} \end{array}$$

Example 3

The equations have no solutions because, during the unification algorithm, we encounter the following contradictory state:

Equations			Substitution		
bool	=	int	t_x	=	bool
t_1	=	int	t_1	=	int
			t_0	=	int

Because bool and int cannot be equal, there is no solution to the equations.

Example 4

$$\underbrace{\texttt{proc}\ (\underbrace{f}_{t_f})\ \underbrace{(\texttt{iszero}\ \underbrace{(f\ f)}_{t_2})}_{t_1}}_{t_0}$$

$$\begin{aligned}
t_0 &= t_f \rightarrow t_1 \\
t_1 &= \text{bool} \\
t_2 &= \text{int} \\
t_f &= t_f \rightarrow t_2
\end{aligned}$$

Example 4

Solving as usual, we encounter a problem:

Equations	Substitution
$\boldsymbol{t_f} = \boldsymbol{t_f} \rightarrow \text{int}$	$\boldsymbol{t_0} = \boldsymbol{t_f} \rightarrow \text{bool}$
	$\boldsymbol{t_1} = \text{bool}$
	$\boldsymbol{t_2} = \text{int}$

- There is no type $\boldsymbol{t_f}$ that satisfies the equation, because the right-hand side of the equation is always larger than the left.
- If we ever deduce an equation of the form $\boldsymbol{t} = \ldots \boldsymbol{t} \ldots$ where the type variable $\boldsymbol{t}$ occurs in the right-hand side, we must conclude that there is no solution. This is called *occurrence check*.

Unification Algorithm

For each equation in turn,

- Apply the current substitution to the equation.
- If the equation is always true (e.g. int $=$ int), discard it.
- If the left- and right-hand sides are contradictory (e.g. bool $=$ int), the algorithm fails.
- If neither side is a variable (e.g. int $\rightarrow \boldsymbol{t_1} = \boldsymbol{t_2} \rightarrow$ bool), simplify the equation, which eventually generates an equation whose left- or right-hand side is a variable.
- If the left-hand side is not a variable, switch the sides.
- If the left-hand side variable occurs in the right-hand side, the algorithm fails.
- Otherwise, move it to the substitution and substitute the right-hand side for each occurrence of the variable in the substitution.

Exercise 1

$$\texttt{let}\ \boldsymbol{x} = \mathbf{4}\ \texttt{in}\ (\boldsymbol{x}\ \mathbf{3})$$

Exercise 2

$$\texttt{let}\ f\ = \texttt{proc}\ (z)\ z\ \texttt{in}\ \texttt{proc}\ (x)\ ((f\ x) - 1)$$

Exercise 3

$$\texttt{let}\ p\ = \texttt{iszero}\ 1\ \texttt{in}\ \texttt{if}\ p\ \texttt{then}\ 88\ \texttt{else}\ 99$$

Exercise 4

$$\texttt{let}\ f\ = \texttt{proc}\ (x)\ x\ \texttt{in}\ \texttt{if}\ (f\ (\texttt{iszero}0))\ \texttt{then}\ (f\ 11)\ \texttt{else}\ (f\ 22)$$

Substitution

Solutions of type equations are represented by substitution:

$$S \in Subst = TyVar \to T$$

Applying a substitution to a type:

$$\begin{array}{rcl} S(\text{int}) & = & \text{int} \\ S(\text{bool}) & = & \text{bool} \\ S(\alpha) & = & \begin{cases} t & \text{if } \alpha \mapsto t \in S \\ \alpha & \text{otherwise} \end{cases} \\ S(T_1 \to T_2) & = & S(T_1) \to S(T_2) \end{array}$$

Example

Applying the substitution

$$S = \{t_1 \mapsto \mathsf{int}, t_2 \mapsto \mathsf{int} \to \mathsf{int}\}$$

to to the type $(t_1 \to t_2) \to (t_3 \to \mathsf{int})$:

$$\begin{aligned}
&S((t_1 \to t_2) \to (t_3 \to \mathsf{int})) \\
&= S(t_1 \to t_2) \to S(t_3 \to \mathsf{int}) \\
&= (S(t_1) \to S(t_2)) \to (S(t_3) \to S(\mathsf{int})) \\
&= (\mathsf{int} \to (\mathsf{int} \to \mathsf{int})) \to (t_3 \to \mathsf{int})
\end{aligned}$$

Unification

Update the current substitution with equality $t_1 \doteq t_2$.

$$\textbf{unify} : T \times T \times Subst \rightarrow Subst$$

$$\begin{array}{rcl}
\textbf{unify}(\text{int}, \text{int}, S) & = & S \\
\textbf{unify}(\text{bool}, \text{bool}, S) & = & S \\
\textbf{unify}(\alpha, \alpha, S) & = & S \\
\textbf{unify}(\alpha, t, S) & = & \begin{cases} \text{fail} & \alpha \text{ occurs in } t \\ \text{extend } S \text{ with } \alpha \doteq t & \text{otherwise} \end{cases} \\
\textbf{unify}(t, \alpha, S) & = & \textbf{unify}(\alpha, t, S) \\
\textbf{unify}(t_1 \rightarrow t_2, t_1' \rightarrow t_2', S) & = & \text{let } S' = \textbf{unify}(t_1, t_1', S) \text{ in} \\
 & & \text{let } S'' = \textbf{unify}(S'(t_2), S'(t_2'), S') \text{ in} \\
 & & \quad S'' \\
\textbf{unify}(\_, \_, \_) & = & \text{fail}
\end{array}$$

Exercises

- $\mathbf{unify}(\alpha, \mathsf{int} \to \mathsf{int}, \emptyset) =$
- $\mathbf{unify}(\alpha, \mathsf{int} \to \alpha, \emptyset) =$
- $\mathbf{unify}(\alpha \to \beta, \mathsf{int} \to \mathsf{int}, \emptyset) =$
- $\mathbf{unify}(\alpha \to \beta, \mathsf{int} \to \alpha, \emptyset) =$

Solving Equations

$$\textbf{unifyall} : \mathit{TyEqn} \rightarrow \mathit{Subst} \rightarrow \mathit{Subst}$$

$$\begin{array}{rcl}\textbf{unifyall}(\emptyset, S) & = & S \\ \textbf{unifyall}((t_1 \doteq t_2) \ \wedge \ u, S) & = & \text{let } S' = \textbf{unify}(S(t_1), S(t_2), S) \\ & & \text{in } \textbf{unifyall}(u, S')\end{array}$$

Let $\mathcal{U}$ be the final unification algorithm:

$$\mathcal{U}(u) = \textbf{unifyall}(u, \emptyset)$$

typeof : $E \to T$

The final type inference algorithm that composes equation derivation ($\mathcal{V}$) and equation solving ($\mathcal{U}$):

$$\begin{array}{l} \textbf{typeof}(E) = \\ \quad \textbf{let } S = \mathcal{U}(\mathcal{V}(\emptyset, E, \alpha)) \quad (\text{new } \alpha) \\ \quad \textbf{in } S(\alpha) \end{array}$$

Examples

- **typeof**$((\texttt{proc}\ (x)\ x)\ 1)$
- **typeof**$(\texttt{let}\ x = 1\ \texttt{in}\ \texttt{proc}(y)\ (x + y))$

Summary

Automatic type inference:

- derive type equations from the program text, and
- solve the equations by unification algorithm.