Specialized Static Analysis Framework: Type Inference

Woosuk Lee

CSE 6049 Program Analysis

Hanyang University, Korea

Some slides are borrowed from http://rightingcode.org/slides/7.pdf

Goal of This Lecture

- Learn practical alternatives to the aforementioned general, abstract interpretation framework

- For simple languages and properties, there are frameworks that are simple yet powerful enough

- But with several limitations

Static Analysis by Proof Construction

- Static analysis = proof construction in a finite proof system
- Finite proof system = a finite set of inference rules for a predefined set of judgements
- The soundness corresponds to the soundness of the proof system
  - The input program is provable => the program satisfies the proven judgement.

Example: Type Inference

- A simple ML-like language

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

Types

Types are defined inductively:

$$\begin{array}{rcl} \boldsymbol{T} & \rightarrow & \textsf{int} \\ & | & \textsf{bool} \\ & | & \boldsymbol{T} \rightarrow \boldsymbol{T} \end{array}$$

Examples:

- int
- bool
- int → int
- bool → int
- int → (int → bool)
- (int → int) → (bool → bool)
- (int → int) → (bool → (bool → int))

Types of Expressions

- Judgement that says expression E has type t is written as

$$\Gamma \vdash e : t$$

- Γ is a set of type assumptions for the free variables in E (called *type environment*)

$$\Gamma : Var \rightarrow T$$

Examples

- $[] \vdash \mathbf{3} : \mathsf{int}$
- $[x \mapsto \mathsf{int}] \vdash x : \mathsf{int}$
- $[] \vdash \mathbf{4} - \mathbf{3} :$
- $[x \mapsto \mathsf{int}] \vdash x - \mathbf{3} :$
- $[] \vdash \texttt{iszero}\ \mathbf{11} :$
- $[] \vdash \texttt{proc}\ (x)\ (x - \mathbf{11}) :$
- $[] \vdash \texttt{proc}\ (x)\ (\texttt{let}\ y = x - \mathbf{11}\ \texttt{in}\ (x - y)) :$
- $[] \vdash \texttt{proc}\ (x)\ (\texttt{if}\ x\ \texttt{then}\ \mathbf{11}\ \texttt{else}\ \mathbf{22}) :$
- $[] \vdash \texttt{proc}\ (x)\ (\texttt{proc}\ (y)\ \texttt{if}\ y\ \texttt{then}\ x\ \texttt{else}\ \mathbf{11}) :$
- $[] \vdash \texttt{proc}\ (f)\ (\texttt{if}\ (f\ \mathbf{3})\ \texttt{then}\ \mathbf{11}\ \texttt{else}\ \mathbf{22}) :$
- $[] \vdash (\texttt{proc}\ (x)\ x)\ \mathbf{1} :$
- $[f \mapsto \mathsf{int} \rightarrow \mathsf{int}] \vdash (f\ (f\ \mathbf{1})) :$

Type System

Inductive rules for assigning types to expressions:

$$\frac{}{\Gamma \vdash n : \mathsf{int}} \qquad \frac{}{\Gamma \vdash x : \Gamma(x)}$$

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

$$\frac{\Gamma \vdash E : \mathsf{int}}{\Gamma \vdash \texttt{iszero}\ E : \mathsf{bool}} \qquad \frac{\Gamma \vdash E_1 : \mathsf{bool} \qquad \Gamma \vdash E_2 : t \qquad \Gamma \vdash E_3 : t}{\Gamma \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 : t}$$

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

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

We say that a closed expression E has type t iff we can derive $[] \vdash E : t$.

Example

Program $\texttt{proc}\ (x)\ (x - \mathbf{11})$ is typed $\mathsf{int} \rightarrow \mathsf{int}$

because we can prove $[] \vdash \texttt{proc}\ (x)\ (x - \mathbf{11}) : \mathsf{int} \rightarrow \mathsf{int}$

as follows:

$$\frac{}{[] \vdash \texttt{proc}\ (x)\ (x - \mathbf{11}) : \mathsf{int} \rightarrow \mathsf{int}}$$

Soundness of Type System

Theorem (Soundness of the proof rules)

Let E be a program, an expression without free variables. If $\emptyset \vdash E : \tau$, then the program runs without a type error and returns a value of type τ if it terminates.

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.

Type Equations

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

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

- Type equations ($\boldsymbol{TyEqn}$) are defined inductively:

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

Example

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

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

Example

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

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

Example

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

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

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

Generating Type Equations

$$
\begin{array}{rcl}
\mathcal{V}(\Gamma, n, t) & = & t \doteq \mathsf{int} \\
\mathcal{V}(\Gamma, x, t) & = & t \doteq \Gamma(x) \\
\mathcal{V}(\Gamma, e_1 + e_2, t) & = & t \doteq \mathsf{int} \ \wedge \ \mathcal{V}(\Gamma, e_1, \mathsf{int}) \ \wedge \ \mathcal{V}(\Gamma, e_2, \mathsf{int}) \\
\mathcal{V}(\Gamma, \texttt{iszero}\ e, t) & = & t \doteq \mathsf{bool} \ \wedge \ \mathcal{V}(\Gamma, e, \mathsf{int}) \\
\mathcal{V}(\Gamma, \texttt{if}\ e_1\ e_2\ e_3, t) & = & \mathcal{V}(\Gamma, e_1, \mathsf{bool}) \ \wedge \ \mathcal{V}(\Gamma, e_2, t) \ \wedge \ \mathcal{V}(\Gamma, e_3, t) \\
\mathcal{V}(\Gamma, \texttt{let}\ x = e_1\ \texttt{in}\ e_2, t) & = & \mathcal{V}(\Gamma, e_1, \alpha) \ \wedge \ \mathcal{V}([x \mapsto \alpha]\Gamma, e_2, t) \ (\mathsf{new}\ \alpha) \\
\mathcal{V}(\Gamma, \texttt{proc}\ (x)\ e, t) & = & t \doteq \alpha_1 \to \alpha_2 \ \wedge \ \mathcal{V}([x \mapsto \alpha_1]\Gamma, e, \alpha_2) \\
 & & (\mathsf{new}\ \alpha_1, \alpha_2) \\
\mathcal{V}(\Gamma, e_1\ e_2, t) & = & \mathcal{V}(\Gamma, e_1, \alpha \to t) \ \wedge \ \mathcal{V}(\Gamma, e_2, \alpha) \ (\mathsf{new}\ \alpha)
\end{array}
$$

Example

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

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	$=$	(int → int) → (int → int)
t_1	$=$	$t_x \to t_2$	t_1	$=$	int → int
t_3	$=$	int	t_2	$=$	int
t_4	$=$	int	t_3	$=$	int
t_2	$=$	int	t_4	$=$	int
t_f	$=$	int $\to t_3$	t_f	$=$	int → int
t_f	$=$	$t_x \to t_4$	t_x	$=$	int

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

Finding a Solution of Type Equations

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

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

Finding a Solution of Type Equations

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

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

Finding a Solution of Type Equations

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

Finding a Solution of Type Equations

Same for the next three equations:

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

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

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

Finding a Solution of Type Equations

Consider the next equation $t_f = \text{int} \to t_3$. 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
$t_f = \text{int} \to \text{int}$	$t_0 = t_f \to (t_x \to \text{int})$
$t_f = t_x \to t_4$	$t_1 = t_x \to \text{int}$
	$t_3 = \text{int}$
	$t_4 = \text{int}$
	$t_2 = \text{int}$

Move the resulting equation to the substitution and update it.

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

Finding a Solution of Type Equations

Apply the substitution to the equation:

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

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

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

Finding a Solution of Type Equations

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

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

The final substitution is the solution of the original equations.

Finding a Solution of Type Equations

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

$$\begin{array}{lcl} t_0 & = & t_f \to t_1 \\ t_1 & = & \text{bool} \\ t_2 & = & \text{int} \\ t_f & = & t_f \to t_2 \end{array}$$

Finding a Solution of Type Equations

Solving as usual, we encounter a problem:

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

- There is no type 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 $t = \ldots t \ldots$ where the type variable 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.

Substitutions

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

$$S = \{t_1 \mapsto \text{int}, t_2 \mapsto \text{int} \rightarrow \text{int}\}$$

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

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

Unification Algorithm

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

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

$$
\begin{array}{rcl}
\textbf{unify}(\mathsf{int}, \mathsf{int}, S) & = & S \\
\textbf{unify}(\mathsf{bool}, \mathsf{bool}, S) & = & S \\
\textbf{unify}(\alpha, \alpha, S) & = & S \\
\textbf{unify}(\alpha, t, S) & = & \begin{cases} \text{fail} & \alpha \text{ occurs in } t \\ \text{extend } S \text{ with } \alpha \doteq t & \text{otherwise} \end{cases} \\
\textbf{unify}(t, \alpha, S) & = & \textbf{unify}(\alpha, t, S) \\
\textbf{unify}(t_1 \rightarrow t_2, t'_1 \rightarrow t'_2, S) & = & \text{let } S' = \textbf{unify}(t_1, t'_1, S) \text{ in} \\
 & & \text{let } S'' = \textbf{unify}(S'(t_2), S'(t'_2), S') \text{ in} \\
 & & \quad S'' \\
\textbf{unify}(\_, \_, \_) & = & \text{fail}
\end{array}
$$

Unification Algorithm

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

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

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

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

Automatic Type Inference

typeof $: E \to T$

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

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

Example

typeof$((\texttt{proc}\ (x)\ x)\ 1)$

Correctness of Automatic Type Inference

Theorem (Correctness of the algorithm)

Solving the equations $\equiv$ proving in the simple type system

Limitations

- For target languages that lack a sound static type system, we have to invent it.
    - Design a finite proof system
    - Prove the soundness of the proof system
    - Design its algorithm that automates proving
    - Prove the correctness of the algorithm
- What if the unification procedure is not enough?
    - For some properties, the algorithm can generate constraints that are unsolvable by the unification procedure
- For some conventional imperative language, sound and precise-enough static type systems are elusive.