Typage

Types

2018-2019
Giovanni Bernardi, gioXYZirif.fr
http://www.irif.fr/~gio/index.xhtml
Université Paris Diderot

Types

Motivations

- Avoid nonsensical programs ($1 + \mathit{true}$)
- Avoid memory violations
- Avoid code whose behaviour is not defined
- Partially specify programs

Plan

- Monomorphic types
  - À la Church
  - À la Curry
    - Unification
    - Type inference
- Polymorphic types
  - À la Church
  - À la Curry (+ type inference)

Monomorphic types à la Church

Expressions à la Church

Types

$$A \quad ::= \quad \mathcal{T} \mid A \times A \mid A \to A$$

$$\mathcal{T} \quad ::= \quad int \mid bool$$

Expressions

$$\begin{array}{lll} M & ::= & x \qquad \mid \\ & & ct \qquad \mid \\ & & \langle M, M \rangle \qquad \mid \\ & & M\ M \qquad \mid \\ & & \lambda x : A.M \qquad \mid \\ & & \mathbf{let}\ x : A = M\ \mathbf{in}\ M \end{array}$$

Few types

$int \to bool$

$bool \times bool$

$bool \to (bool \to int)$

$bool \times (bool \to int)$

$(bool \to bool) \to int$

Few examples

$$\textbf{let } x : \mathit{int} = 3 \textbf{ in } x + 1$$

$$\textbf{let } x : \mathit{int} = (\textbf{if } \mathit{true} \textbf{ then } 1 \textbf{ else } 2) \textbf{ in } x + 1$$

$$\textbf{let } x : \mathit{int} = 4 \textbf{ in } (\textbf{let } y : \mathit{int} = x + 1 \textbf{ in } x * y)$$

$$\textbf{let } f : \mathit{int} \to \mathit{int} = (\lambda x : \mathit{int}.x + 1) \textbf{ in } f\ (f\ x)$$

$$\mathit{fix}(\lambda \mathit{fact} : \mathit{int} \to \mathit{int}.\lambda x : \mathit{int}.\textbf{if } x \textbf{ then } 1 \textbf{ else } (x * \mathit{fact}\ (x - 1)))$$

Reduction semantics

$$
\begin{array}{lcl}
(\lambda x : A.M)\ N & \Rightarrow & M\{x/N\} \\
\textbf{let}\ x : A = N\ \textbf{in}\ M & \Rightarrow & M\{x/N\} \\
\mathit{fix}\ M & \Rightarrow & M\ (\mathit{fix}\ M) \\
\mathit{fst}\langle M, N\rangle & \Rightarrow & M \\
\mathit{snd}\langle M, N\rangle & \Rightarrow & N \\
\textbf{if}\ \mathit{true}\ \textbf{then}\ M\ \textbf{else}\ N & \Rightarrow & M \\
\textbf{if}\ \mathit{false}\ \textbf{then}\ M\ \textbf{else}\ N & \Rightarrow & N \\
\textbf{if}\ 0\ \textbf{then}\ M\ \textbf{else}\ N & \Rightarrow & M \\
\textbf{if}\ n\ \textbf{then}\ M\ \textbf{else}\ N & \Rightarrow & N, \quad n \neq 0
\end{array}
$$

Typing rules à la Church

For every ct there exists a type A, denoted $TC(ct) : A$. A type environement Γ is a set of the form $x_1 : A_1, \ldots, x_n : A_n$.
We write $\Gamma(x_i)$ to denote A_i.

$$\Gamma \vdash x_i : \Gamma(x_i) \qquad \Gamma \vdash ct : TC(ct)$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \to B}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \times B} \qquad \frac{\Gamma \vdash M : A \quad \Gamma, x : A \vdash N : B}{\Gamma \vdash \mathbf{let}\ x : A = M\ \mathbf{in}\ N : B}$$

Examples

Let $(*) = f : \mathit{int} \to \mathit{int} \vdash f : \mathit{int} \to \mathit{int}$

$$\dfrac{\dfrac{x : \mathit{int} \vdash + : \mathit{int} \times \mathit{int} \to \mathit{int} \qquad \dfrac{x : \mathit{int} \vdash x : \mathit{int} \qquad x : \mathit{int} \vdash 1 : \mathit{int}}{x : \mathit{int} \vdash \langle x, 1\rangle : \mathit{int} \times \mathit{int}}}{x : \mathit{int} \vdash +\langle x, 1\rangle : \mathit{int}}}{\color{red}{\emptyset \vdash \lambda x : \mathit{int}. + \langle x, 1\rangle : \mathit{int} \to \mathit{int}}}$$

$$\dfrac{\dfrac{\vdots}{\color{red}{\emptyset \vdash \lambda x : \mathit{int}. + \langle x, 1\rangle : \mathit{int} \to \mathit{int}}} \qquad \dfrac{(*) \qquad \dfrac{(*) \qquad f : \mathit{int} \to \mathit{int} \vdash 3 : \mathit{int}}{f : \mathit{int} \to \mathit{int} \vdash f\ 3 : \mathit{int}}}{f : \mathit{int} \to \mathit{int} \vdash f\ (f\ 3) : \mathit{int}}}{\emptyset \vdash \mathbf{let}\ f : \mathit{int} \to \mathit{int} = (\lambda x : \mathit{int}. + \langle x, 1\rangle)\ \mathbf{in}\ f\ (f\ 3) : \mathit{int}}$$

Properties of the relation $\vdash$

Universal quantifiers are left implicit.

- (Uniqueness): If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A \equiv B$.
- (Weakening): Let $\Gamma = \{x : B \mid x \in FV(M)\}$ and $\Gamma \subseteq \Delta$. We have that $\Gamma \vdash M : A$ iff $\Delta \vdash M : A$.
- (Preservation): If $\Gamma \vdash M : A$ and $M \Rightarrow M'$ then $\Gamma \vdash M' : A$.
- (Subject reduction): If $\Gamma \vdash M : A$ then $M \Rightarrow$ or M normal form.

Typing algorithm

$$
\begin{array}{lll}
\mathit{Type}(\Gamma, ct) & = TC(ct) & \\
\mathit{Type}(\Gamma, x) & = A & \text{if } x : A \in \Gamma \\
\mathit{Type}(\Gamma, \lambda x : A.M) & = A \to B & \text{if } \mathit{Type}((\Gamma, x : A), M) = B \\
\mathit{Type}(\Gamma, \langle M, N \rangle) & = A \times B & \text{if } \mathit{Type}(\Gamma, M) = A \text{ and} \\
 & & \quad \mathit{Type}(\Gamma, N) = B \\
\mathit{Type}(\Gamma, M\ N) & = B & \text{if } \mathit{Type}(\Gamma, M) = A \to B \text{ and} \\
 & & \quad \mathit{Type}(\Gamma, N) = A \\
\mathit{Type}(\Gamma, \mathbf{let}\ x : A = M\ \mathbf{in}\ N) & = B & \text{if } \mathit{Type}(\Gamma, M) = A \text{ and} \\
 & & \quad \mathit{Type}((\Gamma, x : A), N) = B \\
\mathit{Type}(\Gamma, M) & = \mathit{error} & \text{otherwise}
\end{array}
$$

Properties of the algorithm

For every term M and environment Γ,

- (Termination): $\mathit{Type}(\Gamma, M)$ terminates.
- (Soundness): If $\mathit{Type}(\Gamma, M) = A$ then $\Gamma \vdash M : A$.
- (Completeness): If $\Gamma \vdash M : A$ then $\mathit{Type}(\Gamma, M) = A$.

  In other terms,
  if $\mathit{Type}(\Gamma, M) = \mathit{error}$ then M is not typeable in Γ.

Unification theory

Σ-algebras

Σ: Set of function symbols with an arity $n \in \mathbb{N}$.
$\mathcal{X}$: Set of variables.

$\mathcal{T}(\mathcal{X}, \Sigma)$: Set of terms over $\mathcal{X}$ and Σ, inductively defined by

$$\frac{x \in \mathcal{X}}{x \in \mathcal{T}(\mathcal{X}, \Sigma)} \qquad \frac{t_1, \ldots, t_n \in \mathcal{T}(\mathcal{X}, \Sigma) \qquad f \text{ has arity } n \in \Sigma}{f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{X}, \Sigma)}$$

We denote $Var(t)$ the set of all the variables in a term t.
A term t is closed if $Var(t) = \emptyset$.

Substitutions

Definition

- A substitution is a function $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$.
- The domain of a substitution σ is the set $Dom(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$.
- The codomain of a substitution σ is the set $Codom(\sigma) = \{Var(\sigma(x)) \mid x \in Dom(\sigma)\}$.
- A renaming is an injective substitution σ s.t. $\forall x \in Dom(\sigma). \sigma(x) = y$.
  Example: $\sigma = \{x/y, y/w\}$ is arenaming. Every permutation is a renaming, but not the inverse, as shown by the example.
- If the the domain of a substitution σ is finite we denote σ as $\{x_1/t_1, \ldots, x_n/t_n\}$ if $\sigma(x_i) = t_i$ and $x_i \in Dom(\sigma)$.
- The application of a substitution to a term is defined inductively by $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$.

Comparing substitutions

Let σ and τ be two substitutions.
The composition of σ with τ, denoted $\sigma \circ \tau$, is defined as expected by letting $(\sigma \circ \tau)(x) = \sigma(\tau(x))$.

Example

$\{y/b, z/h(c)\} \circ \{x/f(y), y/z\} = \{x/f(b), y/h(c), z/h(c)\}$

Definition (22.4.1 in Pierce book)

The substitution σ is an instance of the substitution τ (or τ is more general than σ), denoted $\sigma \leq \tau$, iff $\exists \rho. \forall x \in \mathcal{X}. \sigma(x) = (\rho \circ \tau)(x)$.

Example

$\{x/f(y), y/z\}$ is more general than $\{x/f(b), y/h(c), z/h(c)\}$

Equivalence for substitutions

The relation $\leq$ is not antisymmetric.[1]

Example

Let $\sigma_1 = \{x/y\}$ and $\sigma_2 = \{y/x\}$. We have that
$\sigma_1 \leq \sigma_2$ as $\sigma_1 = \{x/y\} \circ \sigma_2$, and
$\sigma_2 \leq \sigma_1$ as $\sigma_2 = \{y/x\} \circ \sigma_1$,
but $\sigma_1 \neq \sigma_2$.

Let $\sigma_1 = \{x/y\}$ and $\sigma_3 = \{x/y, z/w, w/z\}$. We have that
$\sigma_1 \leq \sigma_3$ as $\sigma_1 = \{z/w, w/z\} \circ \sigma_3$, and
$\sigma_3 \leq \sigma_1$ as $\sigma_3 = \{z/w, w/z\} \circ \sigma_1$,
but $\sigma_1 \neq \sigma_3$.

Definition

$\sigma \sim \sigma'$ iff $\exists$ renaming ρ s.t. $\sigma = \rho \circ \sigma'$.

And thus for instance $\sigma_1 \sim \sigma_2 \sim \sigma_3$.

---

[1] R antisymmetric if $a\mathcal{R}b$ and $b\mathcal{R}a$ imply $a = b$.

Principal substitution(s)

Let $\mathcal{S}$ be a set of substitutions and let $\tau \in \mathcal{S}$.
We say that τ is principal[2] for $\mathcal{S}$ iff every $\sigma \in \mathcal{S}$ is an instance of τ.

Example

Let $\mathcal{S} = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$, where

$$
\begin{array}{rclrcl}
\sigma_1 & = & \{x/y\} & \sigma_2 & = & \{y/x\} \\
\sigma_3 & = & \{x/y, w/z, z/w\} & \sigma_4 & = & \{x/u, y/u\} \\
\sigma_5 & = & \{x/a, y/a\} & & &
\end{array}
$$

We have that σ_1, σ_2 et σ_3 are principal for $\mathcal{S}$, because

$$
\begin{array}{c}
\sigma_2, \sigma_3, \sigma_4, \sigma_5 \leq \sigma_1 \\
\sigma_1, \sigma_3, \sigma_4, \sigma_5 \leq \sigma_2 \\
\sigma_1, \sigma_2, \sigma_4, \sigma_5 \leq \sigma_3
\end{array}
$$

and $\sigma_1 \not\leq \sigma_4$ and $\sigma_1 \not\leq \sigma_5$.

[2] Also *most general*.

Unifier of a system of equations

Definition

Two terms A and B are unifiable iff there exists a substitution σ s.t. $\sigma(A) = \sigma(B)$ (σ is called a unifier of A and B).

- An equation $A \doteq B$ is formally just a pair of terms, and we say that it is unifiable iff the terms A and B are so.
- A system of equations E is a set of equations. We say that it is unifiable iff there exists one substitution that is the unifier of all the equations in E. This substitution is called solution of E.

We'll focus on *finite* systems of equations.

Example

$f(x, g(x, a))$ and $f(f(a), y)$ unifiable, for instance via $\{x/f(a), y/g(f(a), a)\}$.

$f(x, g(x, a))$ and $f(f(a), f(b, a))$ are not unifiable.

Uniqueness

1. We consider as unifiers of a system E only the substitutions σ s.t. $Dom(\sigma) \subseteq Var(E)$.
2. Let σ and σ' be unifier of a system E. We identify them if they differ only in variable renaming, i.e. if $\sigma \sim \sigma'$.

Example

Let $\mathcal{S} = \{x \doteq y\}$ and let

$$\sigma_1 = \{x/y\}, \quad \sigma_2 = \{y/x\}, \quad \sigma_3 = \{x/y, z/w, w/z\}$$

While $\sigma_1 = \sigma_2$ (for $[\sigma_1]_\sim = [\sigma_2]_\sim$) and they are principal unifiers of $\mathcal{S}$, σ_3 is not considered as a unifier of $\mathcal{S}$.

The principal unifier[3] of a system E is unique up-to renaming, that is: if σ and σ' are two principal unifiers of a system E then $\sigma \sim \sigma'$.

[3] Also called *most general unifier* or *mgu*.

Solved form

Definition

A system of equations E is in solved form iff it has the form $\{\alpha_1 \doteq t_1, \ldots, \alpha_n \doteq t_n\}$, where

- all variables α_i are distinct $\quad (\forall i, j.\, i \neq j \text{ implies } \alpha_i \neq \alpha_j)$
- no α_i appears in a t_j $\quad (\forall i.\, \alpha_i \notin \bigcup_{1 \leq j \leq n} Var(t_j))$

Notation : If E is a system in solved form $\{\alpha_1 \doteq t_1, \ldots, \alpha_n \doteq t_n\}$ we denote $\vec{E}$ the substitution $\{\alpha_1/t_1, \ldots, \alpha_n/t_n\}$.

How to solve equations ?

Manipulation rules

$$\frac{E \cup \{s \doteq s\}}{E} \quad \text{(elimination)} \qquad \frac{E \cup \{t \doteq \alpha\} \quad t \notin \mathcal{X}}{E \cup \{\alpha \doteq t\}} \quad \text{(exchange)}$$

$$\frac{E \cup \{f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)\}}{E \cup \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}} \qquad \text{(decomposition)}$$

$$\frac{E \cup \{\alpha \doteq s\} \quad \alpha \in Var(E) \quad \alpha \notin Var(s)}{E\{\alpha/s\} \cup \{\alpha \doteq s\}} \qquad \text{(replacement)}$$

Unification algorithm

1. The input of the algorithm is a system E
2. The algorithm applies the manipulation rules as long as possible, and computes a system E'
3. If the system E' is in solved form
   - it returns $\vec{E'}$.
   - else it returns Nothing

Example

Let $E = \{f(x, h(b), c) \doteq f(g(y), y, c)\}$.

$$
\cfrac{\cfrac{\cfrac{\cfrac{f(x, h(b), c) \doteq f(g(y), y, c)}{x \doteq g(y), \quad h(b) \doteq y, \quad c \doteq c}\,\mathsf{d}}{x \doteq g(y), \quad h(b) \doteq y}\,\mathsf{e}}{x \doteq g(y), \quad y \doteq h(b)}\,\mathsf{x}}{x \doteq g(h(b)), \quad y \doteq h(b)}\,\mathsf{r}
$$

The principal unifier of E is $\sigma = \{x/g(h(b)), y/h(b)\}$.

Also, $\sigma f(x, h(b), c) = f(g(h(b)), h(b), c) = \sigma f(g(y), y, c)$.

Try exercise 22.4.3 in Pierce book.

Towards the soundness and completeness of the algorithm

Lemma

1. *The unification algorithm terminates.*
2. *If σ is a unifier of a solved form E then $\sigma = \sigma\vec{E}$.*
3. *If a rule transforms a system E into a system E' then the solutions of E and E' are the same.*
4. *If E is in solved form, then $\vec{E}$ is a solution of the system E.*

Proof of termination

A variable is not solved in a system E if it appears in it more than once. The termination of the unification algorithm can be shown reasoning by induction on the triplet $\langle n1, n2, n3 \rangle$ equipped with the lexicographic order, where

n1:	nb of variables not solved
n2:	size of the system
n3:	nb of equation sof the form $t = x$

We have indeed that,

	n1	n2	n3
Remplacement	$>$		
Elimination	$\geq$	$>$	
Decomposition	$=$	$>$	
Exchange	$=$	$=$	$>$

For every finite system of equations E,

Theorem

(Soundness) *If the algorithm executed on E returns a solution $\vec{S}$, then E is unifiable and $\vec{S}$ is a principal unifier for E. If the algorithm returns Nothing then E is not unifiable.*

Theorem

(Completeness) *If E is unifiable, the algorithm returns a principal unifier of E. If the system E is not unifiable then the algorithm returns Nothing.*

Monomorphic types à la Curry

Expressions à la Curry

$$
\begin{array}{rll}
N, M ::= & x & | \\
& cte & | \\
& \langle M, N \rangle & | \\
& M\ N & | \\
& \lambda x.M & | \\
& \mathbf{let}\ x = N\ \mathbf{in}\ M &
\end{array}
$$

Example

$\mathbf{let}\ x : \textit{int} = 3\ \mathbf{in}\ x + 1$ is now $\mathbf{let}\ x = 3\ \mathbf{in}\ x + 1$, and

$\mathbf{let}\ x : \textit{int} = 4\ \mathbf{in}\ (\mathbf{let}\ y : \textit{int} = x + 1\ \mathbf{in}\ x * y))$
is now $\mathbf{let}\ x = 4\ \mathbf{in}\ (\mathbf{let}\ y = x + 1\ \mathbf{in}\ x * y))$

Typing rules à la Curry

$$\Gamma \vdash x : \Gamma(x) \qquad \Gamma \vdash cte : TC(cte)$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \times B} \qquad \frac{\Gamma \vdash M : {\color{red}A} \quad \Gamma, x : {\color{red}A} \vdash N : B}{\Gamma \vdash \mathbf{let}\ x = M\ \mathbf{in}\ N : B}$$

Properties of $\vdash$

Uniqueness is false.

Example

$$\vdash \lambda x.x : int \to int \qquad \vdash \lambda x.x : bool \to bool$$

But the two functions are an instance of a same identitify function that behaves in the same manner in both cases: we have

Polymorphism!

Towards a typing algorithm

- Substitutions
- Principal unifiers of systems of equations
- Unification algorithm for finite systems of equations

- How to build system of equations starting from a program ?
- Typing algorithm via unification

Algorithmic difficulties

$Type(\Gamma, \lambda x.M) = A \to B$ if there exists A s.t
$Type((\Gamma, x : A), M) = B$

$Type(\Gamma, \textbf{let } x = M \textbf{ in } N) = B$ if there exists A s.t.
$Type(\Gamma, M) = A$ and
$Type((\Gamma, x : A), N) = B$

Typing algorithm Inference + unification

Let STC of type schemas for the constants (for example $STC(fst) = \alpha \times \beta \to \alpha$), and let M be a term to be typed.

1. For every variable x of M algo. introduces variable of type α_x, for every subterm N of M algo. introduces variable of type α_N.
2. The algorithm transforms M into a system of equations $SE(M)$ as follows,

M	$SE(M)$
x	$\{\alpha_M \doteq \alpha_x\}$
cte	$\{\alpha_M \doteq STC(cte)\}$
$\langle N, L\rangle$	$\{\alpha_M \doteq \alpha_N \times \alpha_L\} \cup SE(N) \cup SE(L)$
$N\ L$	$\{\alpha_N \doteq \alpha_L \to \alpha_M\} \cup SE(N) \cup SE(L)$
$\lambda x.N$	$\{\alpha_M \doteq \alpha_x \to \alpha_N\} \cup SE(N)$
$\mathbf{let}\ x = N\ \mathbf{in}\ L$	$\{\alpha_M \doteq \alpha_L; \alpha_x \doteq \alpha_N\} \cup SE(N) \cup SE(L)$

3. Unification algorithm solves the system $SE(M)$

Soundness and completeness of the typing algorithm

Rough sketch

Theorem

(Soundness) *If σ is a solution of $SE(M)$, then $\Delta \vdash M : \tau(\alpha_M)$, where $\Delta = \{x : \tau(\alpha_x) \mid x \in FV(M)\}$ and τ is an instance of σ.*

Theorem

(Completeness) *If there exist a Δ and a type A s.t. $\Delta \vdash M : A$, then $SE(M)$ is unifiable.*

Theorem

If there exist a Δ and a type A s.t. $\Delta \vdash M : A$, then A is an instance of $\sigma(\alpha_M)$, where σ is a principal unifier of the system $SE(M)$.