TP Programmation

L3

11 January 2011

In this session we will implement the type inference algorithm for the simply typed λ-calculus. We need to define the abstract syntax of λ-calculus by an ocaml type like
`Term = Var of String | Apl of Term * Term | Fun of String * Term`

Given a λ-term, we need to find the most general type of it. We will achieve this by reducing it to the unification problem. We will generate a set of equations E such that the most general type of the λ-term is given by the most general unifier of E. Let us recall what we saw in class.

- A goal is a finite set G of triples (Γ, M, τ) where Γ is a context, M a λ-term, and τ a simple type.
- We assume that all bound variables in M are distinct, that all free variables occur in the context Γ, and that for every variable x we have a type variable t_x.
- We define a reduction relation on pairs (G, E).
- Assuming $G = \{g\} \cup G'$ and $g \equiv (\Gamma, M, \tau) \notin G'$ all the rules produce a pair $(G' \cup G_g, E \cup E_g)$ where G_g and E_g are defined as follows:

g	G_g	E_g	
(Γ, x, τ)	$\emptyset$	$\{t_x = \tau\}$	
$(\Gamma, M_1 M_2, \tau)$	$\{(\Gamma, M_1, t_1 \to \tau), (\Gamma, M_2, t_1)\}$	$\emptyset$	t_1 fresh
$(\Gamma, \lambda x.M_1, \tau)$	$\{(\Gamma, x : t_x, M_1, t)\}$	$\{\tau = t_x \to t\}$	t fresh

- Given a term M_0 with free variables $x_1, \ldots, x_n$ we set the initial pair to $(G_0, \emptyset)$ with $G_0 = \{(\Gamma_0, M_0, t_0)\}$ and $\Gamma_0 = x_1 : t_{x_1}, \ldots, x_n : t_{x_n}$.
- Let E_f be the set of equations we derived from the initial goal. If we take the most general unifier S of E_f and we apply it to t_0, we obtain the most general type of M_0.

You need to include this TP also in the report which is to be submitted on 14th. Please include the documented code of this TP and the input and output to find the most general type of $\lambda k.(k(\lambda x.\lambda h.hx))$.