Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lukas Stevens

Lambda Calculus
Winter Term 2021/22
Solutions to Exercise Sheet 8

Exercise 1 (Type Inference in Haskell)

In this exercise, we will develop a type inference algorithm for the simply typed λ-calculus in Haskell. The general idea of the algorithm is to apply the type inference rules in a backward manner and to record equality constraints between types on the way. These constraints are then solved to obtain the result type.

a) Take a look at the template provided on the website. We have provided definitions of terms and types in the simply typed λ-calculus, together with syntax sugar for input and printing. Moreover, you can find the type of substitutions and utility functions to work with substitutions, types and terms.

b) The first component of the algorithm is *unification* on types. Given a list of equality constraints between types of the form $u_1 \stackrel{?}{=} t_1, \ldots, u_n \stackrel{?}{=} t_n$, we want to produce a suitable substitution ϕ such that $\phi(u_i) = t_i$ for all $1 \leq i \leq n$ or report that the given constraints do not have a solution. Fill in the remaining cases of the function *solve* that achieves this functionality.

c) Now we want to apply the type inference rules and record the arising type constraints. Function *constraints* of type

$$\mathit{Term} \to \mathit{Type} \to \mathit{Env} \to \mathit{(Int, [(Type, Type)])} \to \mathit{Maybe\ (Int, [(Type, Type)])}$$

will achieve this functionality. Given a term t, a type τ, an enviroment Γ, and a pair (n, C), it will try to justify $\Gamma \vdash t \colon \tau$, adding the arising type constraints to C. The natural number n is used to keep track of the least variable index that is currently unused. This allows to easily generate fresh variable names. Complete the definition of *constraints*.

d) Define the function *infer* that infers the type of a term by combining *solve* and *constraints* and try it on a few examples.

Solution

See *type_inference.hs*.

Exercise 2 (Every Type is Applicative)

a) Show that every type is *substitutive*.

b) Show that every type is *applicative*.

Solution

a) We first show that every type τ is of the form

$$\tau_1 \to \dots \tau_n \to \tau'$$

with τ' not of function type by induction on τ. The case where τ is elementary is immediate. If $\tau = \tau_1 \to \tau_2$, τ_2 is either not of function type and we are done, or we can apply the induction hypothesis to τ_2, and we are done. Note that $\to$ associates to the right.

Now, we use this as an induction rule on types to show the original statement. Thus, assume $\tau = \tau_1 \to \dots \tau_n \to \tau'$, and that the τ_i are all substitutive (IH). By Lemma 3.2.3, the τ_i are all applicative, and thus τ is substitutive by Lemma 3.2.4.

b) By Lemma 3.2.3

Exercise 3 (Alternative Proof of Lemma 3.2.3)

a) Show that for every $s \in T$, $s\,x \in T$ if x is fresh with respect to s.

b) Show that every substitutive type is *applicative*.

Solution

a) By induction on $s \in T$.

Case Var: Then $s = y\, r_1\ \dots\ r_n$ with $r_1, \dots, r_n \in T$ for some variable y. Since $x \in T$ by rule Var, we get $y\ r_1\ \dots\ r_n\ x \in T$ by rule Var.

Case λ: We assume that $s = (\lambda y.\ t)$ and $t \in T$ and x is fresh w.r.t y and t. We want to use the rule β to prove $(\lambda y.\ t)\ x \in T$ which means that we need to prove $t[x/y] \in T$. The proof is by another induction on $t \in T$.

Case Var: Then $t = z\ r_1\ \dots r_n$ with $r_1, \dots, r_n \in T$. We get the induction hypotheses that $r_1[x/y], \dots, r_n[x/y] \in T$. Additionally, we have $y \in T$ and $x \in T$ by rule Var. Thus, we have that $t[x/y] = z[x/y]\ r_1[x/y]\ \dots\ r_n[x/y]\ x[x/y] = z[x/y]\ r_1[x/y]\ \dots r_n[x/y]\ y \in T$.

Case β: Then $t = (\lambda z.\ r)$ and $r \in T$ with the IH $r[x/y] \in T$. Thus, $t[x/y] = (\lambda z.\ r)[x/y] = (\lambda z.\ r[x/y])$ due the the freshness of x. Now we can use the rule λ to conclude that $t[x/y] \in T$.

Case λ: Then $t = (\lambda z.\ r)\ u\ u_1\ \dots\ u_n$ and $r[u/z]\ u_1\ \dots\ u_n \in T$ and $u \in T$. As IH we get that $(r[u/z]\ r_1\ \dots\ r_n)[x/y] \in T$ and $u[x/y] \in T$. From this we have $(r[u/z]\ r_1\ \dots\ r_n)[x/y] = r[x/y][u[x/y]/z]\ r_1[x/y]\ \dots\ r_n[x/y] \in T$ by Lemma 1.1.5 using the fact that x is fresh w.r.t. both u and y.

Case β: We have $s = r[t/x]\ t_1 \dots t_n\ x \in T$ by IH and $t \in T$ as another precondition, and thus can directly apply β.

b) Assume that τ is substitutive. Further assume that we have $t\colon \tau \to \sigma$, $r\colon \tau$, $t \in T$, and $r \in T$. With the part a) we get that $t\ x \in T$ for fresh $x : \tau$. Because τ is substitutive, we have

$$t\,r = (t\ x)[r/x] \in T\ .$$

Homework 4 (Types of Church Numerals)

a) Let τ be any type. Show that for the n-th Church numeral $\underline{\mathrm{n}}$, we have

$$[] \vdash \underline{\mathrm{n}} \colon (\tau \to \tau) \to \tau \to \tau$$

.

b) Show that every term $t \in \mathrm{NF}$ with $[] \vdash t \colon (\iota \to \iota) \to \iota \to \iota$, t is either id or a church numeral. Here ι is any *elementary* type.

Homework 5 (Completeness of T)

In this exercise, you will show the converse of Lemma 3.2.2, i.e.

$$\Downarrow t \Longrightarrow t \in T$$

.

a) Show that every λ-term has one of the following shapes:

- $x\, r_1 \ldots r_n$
- $\lambda x.\ r$
- $(\lambda x.\ r)\, s\, s_1\ \ldots\ s_n$

Note that this gives rise to an alternative inductive definition for λ-terms and to a corresponding rule induction on λ-terms.

b) $\Downarrow$ gives rise to a wellfounded induction principle. To show

$$\forall t.\ \Downarrow t \Longrightarrow P(t)\,,$$

it suffices to prove:

$$\forall t.\ (\forall t'.\ t \to_\beta t' \Longrightarrow P(t')) \Longrightarrow P(t)\,.$$

Use this to prove:

$$\Downarrow t \Longrightarrow t \in T$$

Hint: Use (a) for an inner induction on terms.