Proofs and Programs

Week 5, Tutorial 5 - Polymorphism

Philippe Audebaud, Aurore Alcolei

Thursday, 8th March 2018 — **HW** due before Tuesday, 13th March, 8h00 **hard dead line**

Goals (Weeks 5 & 6) : • Typability and inhabitation for System F (alias $\lambda 2$). • Expressivity power allowed by polymorphism. • In particular, representations of both Propositional Calculus (Logic) and common Free Structures. • Major meta-properties and proof techniques. – *Eventually, at the end of week 6, being able to play both sides of Curry-Howard correspondence (statics + dynamic), and being able to extrapolate this understanding to higher-order types systems is at stake.*

Notation Inference rules for System F *à la Church* are given in appendix.

Exercice 1 (Normal Forms)**.** We admit that well-formed terms in System F are strongly normalising (see lecture 6). Reduce the following term to its normal form (step by step):

$$(\lambda b^{\forall X, X \to X \to X}.(\Lambda X.b\ (\forall X.X \to X \to X\ (\Lambda X.\lambda x^X y^X.y)\ (\Lambda X.\lambda x^X y^X.x)))(\Lambda X.\lambda x^X y^X.y)$$

Show that normal forms can be defined using a BNF grammar. (Hint: recall the one from $\lambda_{\to}$.)

Exercice 2 (terms and types new relationship)**.** Assuming that $X \notin \mathrm{FV}(A)$ and $Y \notin \mathrm{FV}(B)$. Solve the inhabitation problem $(\vdash ? : T)$ when type T is:

a) $A \to (\forall X.(A \to X) \to X)$; $(\forall X.(A \to X) \to X) \to A$;

b) **HW** $\forall Z \forall Y.((\forall X.(X \to Z)) \to Y \to Z)$; $((\forall Y.A) \to (\forall X.B)) \to (\forall X, Y.A \to B)$.

Exercice 3 (Type inference)**.** Study the type inference problem $(\vdash t : ?)$ when term t is:

a) $\Lambda X.\lambda f^{X \to X}.\lambda x^X.f\ \ (f\ x)$;
$\Lambda Y.\lambda x^{\forall X.(X \to X)}.x\ \ (Y \to X)\ \ (x\ Y)$;

b) **HW** $\lambda f^{\forall X.(X \to T \to X)}.\Lambda Y.\lambda x^Y.f\ \ (T \to Y)\ \ (f\ Y\ x)$.

Starting from the following pure lambda-terms, which are therefore almost never well-formed in system F *à la Church*, find whenever possible, a type "decoration" and a " most general" type in system F :

a) $\mathsf{I} \equiv \lambda x.x$, $\mathsf{T} \equiv \lambda x.\lambda y.x$, $\mathsf{F} \equiv \lambda x.\lambda y.y$;

b) from previous point, propose a coding for the type **bool** of booleans in system F. Complete with the conditional **if** (cf. tutoral 1) ;

c) **HW** Let $e \equiv (\lambda y.\lambda z.z\ \ (y\ \mathsf{I})\ \ (y\ \mathsf{F}))\ \ \boldsymbol{\Delta}$, a pure λ-term. *(i)* Is it strongly normalising? *(ii)* Is it possible to assign a type to e, in $\lambda_{\to}$? *(iii)* Is it possible to provide a decoration $\hat{e}$ for e, as a well-formed term in system F? Eventually build the full derivation tree leading to $\hat{e}$.

Exercice 4 (Product)**.** By taking advantage of both the results from tutorial 1, and the previous analysis of booleans, find a proper representation for the general product $A \times B$ of types A and B.
Since is $\top$ (True) is a "limit case", deduce its proper representation in system F.

Exercice 5 (Sum)**.** Do the same for the sum (co-product) $A + B$ of types A and B and $\bot$ (False).

Exercice 6 (Logic encoding)**.** Take advantage of tutorial 4 to provide a complete representation of the propositional calculus NJ in system F.

Exercice 7 (Church integers)**.** For *some* reason, the correct representation for Church integers in system F starts with the polymorphic type **nat** $\equiv \forall X.X \to (X \to X) \to X$.

a) In the light of previous exercises, explain this definition.

b) Provide a representation for each natural number representative $\bar{n}$: **nat**, where $n \in \mathbb{N}$.

c) define zero $\mathbf{Z}$ and the successor function $\mathbf{S}$. What would be the corresponding introduction rules for **nat** related to them?

d) Propose an abstract elimination rule for **nat**, and show the existence of a well-formed term, in system F, that codes for this elimination rule.

e) We want to offer the **iteration** schema, along the following abstract equalities :

$$\mathbf{iter}\ x\ f\ \mathbf{Z} = x \quad \text{and} \quad \mathbf{iter}\ x\ f\ (\mathbf{S}\,p) = f\ (\mathbf{iter}\ x\ f\ p)$$

Show that **iter** is representable in system F. Is it true that for all $n \in \mathbb{N}$, **iter** $x\ f\ \overline{n+1}$ reduces to f (**iter** $x\ f\ \bar{n}$)?

f) **HW** Complete with the proper coding of both **add** and **pred** in system F.

g) **HW** We want to offer the even more powerful **recursion** schema. It should obey the abstract equalities:

$$\mathbf{R}\ x\ f\ \bar{0} = x \quad \text{and} \quad \mathbf{R}\ x\ f\ \overline{n+1} = f\ (\mathbf{R}\ x\ f\ \bar{n})\ \bar{n}$$

Show off your skills!

A System F "*à la Church*"

Types can still be represented with the help of a BNF grammar ($\forall$ is dominant over $\to$):

$$\text{(types)} \qquad T \quad ::= \quad X \in \mathcal{V} \mid T \to T \mid \forall X.T$$

Pre-terms can also be described this way, but they do not necessarily correspond to well-formed terms

$$\text{(terms)} \qquad t \quad ::= \quad x \in \mathcal{X} \mid \lambda x^T.t \mid t\ t \mid \Lambda T.t \mid t\ T$$

A **typing context** is an unordered list: $\Delta \equiv x_1 : T_1, \ldots, x_n : T_n$, st each term variable occurs only once. The notation $\Delta \vdash_{\lambda 2} t : T$ stands for any **judgement** which can be built upon the following inference system:

$$\text{(Hyp)}\ \frac{x : T \in \Delta}{\Delta \vdash x : T} \qquad (\to I)\ \frac{\Delta, x : S \vdash t : T}{\Delta \vdash \lambda x^S.t : S \to T} \qquad \frac{\Delta \vdash e : S \to T \quad \Delta \vdash s : S}{\Delta \vdash e\ s : T}\ (\to E)$$

$$(\forall I)\ \frac{\Delta \vdash t : T \quad X \notin \mathrm{FV}(\Delta)}{\Delta \vdash \Lambda X.t : \forall X.T} \qquad \frac{\Delta \vdash t : \forall X.T}{\Delta \vdash t\ S : T\langle S/X\rangle}\ (\forall E)$$

In particular, a pre-term t is **well-formed** iff there exists a context Δ, and a type T such that $\Delta \vdash_{\lambda 2} t : T$.

Reductions in System F are defined upon the two following steps:

$$(\lambda_x.t)s \to_\beta t\langle s/x\rangle \qquad\qquad (\Lambda X.t)T \to_B t\langle T/X\rangle$$