Proofs and Programs

Week 6, Tutorial 6 - Encodings in System F

Philippe Audebaud, Aurore Alcolei

Thursday, 15th March 2018 — **HW** due before Tuesday, 20th March, 8h00 **hard dead line**

Goal : Explore the expressivity power allowed by polymorphism, via the encodings of several free structures.

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

Exercice 1. (Warming up)

1. Recall the encodings of pairs $A \times B$ and sum $A + B$ in system F and their main constructors/destructors. Explain informally why this constructions make sense.

2. For $\tau(Y)$ a type in system F with some free type variable Y, we say that $\tau(Y)$ is *(weakly) functorial* if there exists a term F in normal form such that

$$\vdash F : \forall Y_1, Y_2, (Y_1 \to Y_2) \to \tau(Y_1) \to \tau(Y_2)$$

   Show that the encodings of $A + B$ and $A \times B$ are weakly functorial by varying one of their two variables.

Exercice 2 (Lists, Trees, and more)**.** In this exercise we want to capture a general recipe to build free structures in System F. In general, a structure Θ is generated by a finite number of *constructors* $c_1, \cdots, c_n$, of respective type S_i such that

$$S_i \equiv T_1^i \to \cdots \to T_{k_i}^i \to \Theta$$

where Θ can only occurs *positively* (*ie.* left to an even number of arrows) in the T_j^i's.

Good type candidates for encoding such structures must at least provide a way to encode their constructors and a way to "destruct" them, that is, a way to define functions by induction on the structure of Θ. Starting from some type U and functions $g_1, \cdots, g_n$ of types $S_i\langle U/\Theta\rangle$ this amounts to be able to define a function $h : \Theta \to U$ such that

$$h(c_i\ x_1 \cdots x_{k_i}) = g_i(hx_1) \cdots (hx_{i_k})$$

1. Explain why the type $T \equiv \forall X. S_1\langle X/\Theta\rangle \to \cdots \to S_n\langle X/\Theta\rangle \to X$ is a good candidate to encode Θ according to the above criteria.

2. Using the previous recipe, give an encoding for lists of element of type A (express it first as a free structure!). Can you define a map function using this encoding?

3. **HW** Same question for trees of elements of type B.

Exercice 3 (Church integers)**.** For *some* reason, the correct representation for Church integers in system F starts with the polymorphic type $\textbf{nat} \equiv \forall X. X \to (X \to X) \to X$.

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

b) Provide a representation $\bar{n} : \textbf{nat}$ for each natural number $n \in \mathbb{N}$. Show that these are actually the only terms in normal form inhabiting **nat**. Is that true if $\textbf{nat} \equiv \forall X. (X \to X) \to X \to X$?

c) Define zero Z and the successor function 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 :

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

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

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:

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

Exercice 4 (Algebraic datatypes)**.** In previous exercises we have seen several examples of encodings of free structures in system F. In this exercise we study an other "general recipe" for encoding *algebraic datatypes*. Algebraic datatypes are those that can be defined by an equation $\Theta = A(\Theta)$ where A is described with sums and products.

The idea will be to choose as representative for these types, the type of system F that corresponds to the *initial algebra* for $A(\Theta)$. In other word, we will define T the representative type of Θ such that T has a *principle of induction* $I : A(T) \to T$ (ie T is an *algebra*) and T is the most general possible types that enjoys this property (T is *initial*).

1. Give an algebraic definition for lists of elements of type A. Can you generalise this construction to any free structures?

2. In Exercise 1 we saw that the encodings of $A \times B$ and $A + B$ are functorial. More generally, prove that any algebraic type $A(T)$ is functorial.

3. For $A(X)$ a functorial type, let $\iota \equiv \forall X, (A(X) \to X) \to X$. Find two closed terms, $I : A\langle \iota / X \rangle \to \iota$ and $R : \forall X, (A(X) \to X) \to \iota \to X$ such that for every $t : A\langle B/X \rangle \to B$

$$(R\ B\ t)(I\ x) \to_\beta^* f(A\ \iota\ B(R\ B\ f)x))$$

4. For $A(X)$ a functorial type, we call A-*algebra* the data of a type T and a term t such that $\vdash t : A\langle T/X \rangle \to A$. We say that a A-algebra (T, t) is *initial* if for every T-algebra (T', t') there exists a (unique) term (in normal form) $s : T \to T'$, such that $\lambda x^{A\langle T/X \rangle}.s\ (t\ x) =_\beta \lambda x^{A\langle T/X \rangle}.t'\ ((F\ T\ T'\ s)\ x)$.

   Explain why being an initial A-algebra formalize the idea of being the most general encoding of an algebraic type $A(X)$. Show that (ι, I) is an initial T-algebra.

A System F "*à la Church*"

Types $T \quad ::= \quad X \in \mathcal{V} \mid T \to T \mid \forall X.T$

Pre-terms $t \quad ::= \quad x \in \mathcal{X} \mid \lambda x^T.t \mid t\ t \mid \Lambda T.t \mid t\ T$

Typing rules

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

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

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

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