Logic and Programs 2000:Curry-Howard isomorphism

Yasugi: 林晋著（コロナ社）数理論理学より

平成 15 年 6 月 28 日

1 Functional programs with types

We are to define a (theoretical) typed programming language F_0. A type means a type of data. There can be multiple atomic types.

Definition 1.1 (Types) 1) We prepare type variables, $\alpha_1, \alpha_2, \cdots, \alpha_n, \cdots$ (abbreviated to $\alpha, \beta, \gamma, \cdots$)

A type variable is a *type*.

2) If σ_1, σ_2 are types, then $(\sigma_1 \times \sigma_2)$ and $(\sigma_1 \rightarrow \sigma_2)$ are types.

Definition 1.2 (Program of F_0) The programs of F_0 and their types are defined as follows. We prepare program variables $x_1, x_2, \cdots, x_n, \cdots$ (abbreviated to $x, y, z, \cdots$), and define fee and bound occurrences of variables in programs. An expression $e : \sigma$ will denote the declaration that "e is a program of type σ".

When $x : \sigma$ is declared, we say that this expression is called a typed variable.

When $e : \sigma$ is declared, we say that this expression is called a typed program.

A set of typed variables, say S is called consistent if the condition that, for any two declaration $x : \sigma$ and $x : \tau$ with a same variable x in S, $\sigma = \tau$.

1) For each variable x and each type σ, $x : \sigma$ will declare that x is a program of type σ. x occurs free in $x : \sigma$. (Atomic programs)

The following are compound programs.

2) Suppose $e : \sigma_1 \times \sigma_2$. Then $fst(e) : \sigma_1$ and $snd(e) : \sigma_2$.

(Projection to the first and to the second type respectively)

A variable x in $fst(e) : \sigma_1$ occurs free (bound) if x occurs free (bound) in e. Similarly with $snd(e) : \sigma_2$.

3) Suppose $e_1 : \sigma_1$ and $e_2 : \sigma_2$. Let S_1 be the set of typed variables occurring free in $e_1 : \sigma_1$ and let S_2 be the set of typed variables occurring free in $e_2 : \sigma_2$. Suppose $S_1 \cup S_2$ is consistent. Then $\langle e_1, e_2 \rangle : \sigma_1 \times \sigma_2$. A typed variable x in this program occurs free (bound) if it occurs free (bound) in either $e_1 : \sigma_1$ or $e_2 : \sigma_2$. The set of variable occurring free is $S_1 \cup S_2$.

(Program of direct product type)

4) Suppose $e : \sigma$, and let the set of variables in this declaration be S. Suppose $S \cup \{x : \tau\}$ is consistent. Then $(funx.e) : \tau \to \sigma$. All occurrences of x in this declaration are bound (by $funx$). For any other occurrence of a variable, it occurs free (bound) if it *is* in $e : \sigma$.

(Functional type program)

5) Suppose $e_1 : \tau \to \sigma$ and $e_2 : \tau$. Assume further that the union of the sets of free occurrences of variables in both declarations is consistent. Then $e_1(e_2) : \sigma$. A variable occurs free (bound) in this declaration occurs free either in $e_1 : \sigma \to \tau$ or $e_2 : \sigma$. The set of variables occurring free is the union of such for $e_1 : \sigma \to \tau$ and $e_2 : \sigma$.

(Application type program)

Note 1) We sometimes abbreviate types in a program if no confusion is anticipated. In fact, if we assign types to variables, then types of all the compound programs are automatically determined. We will also abbreviate parentheses when no confucion is anticipated.

2) 2) and 3) make a pair. 2) executes a computation, while 3) composes a program. 4) and 5) make a pair. 4) composes a program while 5) executes a computation.

Definition 1.3 Reduction (computation) of programs. 1) One step computation.

$$fst\,\langle e_1, e_2\rangle \hookrightarrow e_1$$

$$snd\,\langle e_1, e_2\rangle \hookrightarrow e_2$$

$$(funx.e_1)(e_2) \hookrightarrow e_1[e_2/x]$$

where in general $t[s/x]$ denotes the result of substituting s for the variable x in t.

2) A sequence of computations. If t is obtained from s by a repetition of reductions (a sequence of computations):

$$s \hookrightarrow s_1 \hookrightarrow s_2 \hookrightarrow \cdots \hookrightarrow t,$$

then write $s \hookrightarrow\!\!\hookrightarrow t$.

3) t is normal if no reduction (computation) applies to t, namely, the computation halts at t.

Proposition 1.1 Every computation of a term (program) s of F_0 halts (with a unique normal t). t is called the value of s.

2 Programs and proofs

Programs of F_0 correspond to "some" proofs of the propositional logic with $\wedge$ and $\Rightarrow$ only.

Proposition 2.1 1) A type and a propositional formula corresponds each other.

$X_i \leftrightarrow \alpha_i$: (The ith propositional variable and the ith type variables correspond each other.)

$A \wedge B \leftrightarrow \sigma \times \tau$: ($\wedge$ and $\times$ correspond each other.)

$A \Rightarrow B \leftrightarrow \sigma \rightarrow \tau$: ($\Rightarrow$ and $\rightarrow$ correspond each other.)

2) A term construct and an admissible inference correspond each other.

$e_i \leftrightarrow \Pi_i, i = 1, 2 \longrightarrow \langle e_1, e_2\rangle \leftrightarrow$

$$\frac{\Pi_1 \quad \Pi_2}{A_1 \wedge A_2}$$

, where Π_i is a proof of A_i.

$x \leftrightarrow [x : A], e \leftrightarrow \Pi \longrightarrow funx.e \leftrightarrow$

$$\frac{\Pi}{A \Rightarrow B}$$

where Π is a proof of B from the assumption $[x : A]$ with label x.