LLLang reference

Jean Caspar

3 février 2023

Table des matières

1 Grammar

x is a identifier and n an integer.

$$
\begin{array}{rcl}
A, B, C, \ldots & ::= & \texttt{int} \\
& | & x \\
& | & x\langle A, B, \ldots \rangle \\
& | & () \\
& | & ! \\
& | & !A \\
& | & A * B * \ldots \\
& | & A + B + \ldots \\
& | & A \multimap B \\
& | & \mu x.A \\
& | & \forall x.A
\end{array}
\qquad
\begin{array}{rcl}
P, Q, \ldots & ::= & \_ \\
& | & x \\
& | & n \\
& | & () \\
& | & P, Q, \ldots \\
& | & \texttt{inj}\ n\ P
\end{array}
$$

$$
\begin{array}{rcll}
e, f, g, \ldots & ::= & x & | n \\
& & | () & | e\langle A, B, \ldots \rangle \\
& & | \texttt{roll}\ A\ e & | \texttt{unroll}\ e \\
& & | \texttt{inj}\ A\ n\ e & | e\ f \\
& & | \texttt{let}\ P = e\ \texttt{in}\ f & | \texttt{let}\ !x = e\ \texttt{in}\ f \\
& & | -e & \\
& & | e + f & | e - f \\
& & | e * f & | e/f \\
& & | e\%f & | e = f \\
& & | e < f & | e, f, \ldots \\
& & | \texttt{match}\ e\ \{P \Rightarrow f, Q \Rightarrow g, \ldots\} & \\
& & | \texttt{fun}\langle x, y, \ldots \rangle (P : A, Q : B, \ldots) \multimap C\{e\} & \\
& & | \texttt{rec fun}\langle x, y, \ldots \rangle (P : A, Q : B, \ldots) \multimap C\{e\} &
\end{array}
$$

Currently, in the syntaxe $x\langle A, B, \ldots, \rangle$, x should be a named type, and not a type variable. Furthermore, x should be the name of a type of the form $\forall y_1 \ldots \forall y_n.T$ in order for $x\langle A_1, \ldots, A_n \rangle$ to be a type.

2 Typing

2.1 Subtyping

Subtyping is the relation $A <: B$, ich means that A can be used wherever B is needed. It is the smallest preorder that satisfies the following relations :

$$\frac{}{! <: A}$$

$$\frac{A' <: A \quad B <: B'}{A \multimap B <: A' \multimap B'}$$

$$\frac{\forall 1 \leq i \leq n \;\; A_i <: B_i}{A_1 * \cdots * A_n <: B_1 * \cdots * B_n}$$

$$\frac{\forall 1 \leq i \leq n \;\; A_i <: B_i}{A_1 + \cdots + A_n <: B_1 + \cdots + B_n}$$

$$\frac{A <: B}{\mu x.A <: \mu x.B}$$

$$\frac{A <: B}{\forall x.A <: \forall x.B}$$

2.2 Pattern typing

We say that a pattern P can match a type T and bind variables $x_1 : T_1, \ldots, x_n : T_n$ if one can derives $x_1 : T_1, \ldots, x_n : T_1 \vdash P \prec T$ from the following relations :

$$\frac{}{\vdash \_ \prec T}$$

$$\frac{}{x : T \vdash x \prec T}$$

$$\frac{}{\vdash n \prec \texttt{int}}$$

$$\frac{}{\vdash () \prec ()}$$

$$\frac{x_1 : T_1, \ldots, x_n : T_n \vdash P \prec T}{x_1 :! T_1, \ldots, x_n :! T_n \vdash P \prec ! T}$$

$$\frac{\forall 1 \leq i \leq n \;\; x_{i,1}, \ldots, x_{i,n_i} \vdash P_i \prec T_i \quad \forall 1 \leq i < j \leq n \;\; \{x_{i,k} \mid 1 \leq k \leq n_i\} \cap \{x_{j,k} \mid 1 \leq k \leq n_j\} = \varnothing}{x_{1,1} : T_{1,1}, \ldots, x_{1,n_1} : T_{1,n_1}, \ldots, x_{k,n_k} : T_{k,n_k} \vdash P_1, \ldots, P_k \prec T_1 * \cdots * T_k}$$

$$\frac{x_1 : T_1, \ldots, x_k : T_n \vdash P \prec A_i \quad 1 \leq i \leq n}{x_1, \ldots, x_k \vdash \texttt{inj}\ i\ P \prec A_1 + \cdots + A_n}$$

2.3 Irrefutable patterns

A pattern is said to be irrefutable if it cannot fail to match. Such patterns are :
— Discarding
— Binding to a variable
— A tuple of irrefutable patterns (the empty tuple is such a tuple)
— An injection of a sum type of size 1 and an irrefutable pattern inside (this should probably not happen)

In let bindings or in function arguments, patterns that appear should be irrefutable.

2.4 Pattern matching