Typify types

Oleg Grenrus

August 4, 2014

Parser and tools for type signatures of typify. Essentially the language is dependent type theory, with omitted lambda constructor.

1 Formal syntax definition

Terminals:

$$\frac{}{\top : \mathsf{type}}\ \boxed{\textsc{Any}} \qquad \frac{}{\bot : \mathsf{type}}\ \boxed{\textsc{Nothing}} \qquad \frac{}{\mathbb{1} : \mathsf{type}}\ \boxed{\textsc{Unit}}$$

$$\frac{}{\texttt{number} : \mathsf{type}}\ \boxed{\textsc{Number}} \qquad \frac{}{\texttt{string} : \mathsf{type}}\ \boxed{\textsc{String}} \qquad \frac{}{\texttt{bool} : \mathsf{type}}\ \boxed{\textsc{Boolean}}$$

Identifiers:

$$\frac{n : \mathsf{name}}{n : \mathsf{type}}\ \boxed{\textsc{Identifier}}$$

Record pairs:

$$\frac{n : \mathsf{name} \quad a : \mathsf{type}}{n : a : \mathsf{pair}}\ \boxed{\textsc{Record-Pair}} \qquad \frac{r : \mathsf{pair}}{r : \mathsf{rlist}}\ \boxed{\textsc{Record-Singleton}} \qquad \frac{p : \mathsf{pair} \quad r : \mathsf{rlist}}{p; r : \mathsf{rlist}}\ \boxed{\textsc{Record-Cons}}$$

Records:

$$\frac{}{\{\} : \mathsf{type}}\ \boxed{\textsc{Empty-Record}} \qquad \frac{r : \mathsf{rlist}}{\{r\} : \mathsf{type}}\ \boxed{\textsc{Record}}$$

Row types:

$$\frac{a : \mathsf{type} \quad b : \mathsf{type}}{a \wedge b : \mathsf{type}}\ \boxed{\textsc{Conjunction}} \qquad \frac{a : \mathsf{type} \quad b : \mathsf{type}}{a \vee b : \mathsf{type}}\ \boxed{\textsc{Disjunction}} \qquad \frac{a : \mathsf{type}}{a? : \mathsf{type}}\ \boxed{\textsc{Optional}}$$

Application:

$$\frac{a : \mathsf{type} \qquad b : \mathsf{type}}{a\ b : \mathsf{type}}\ \boxed{\textsc{Application}}$$

Special modifiers:

$$\frac{a : \mathsf{type} \quad b : \mathsf{type}}{a \times b : \mathsf{type}}\ \boxed{\textsc{Product}} \qquad \frac{a : \mathsf{type}}{a \ldots : \mathsf{type}}\ \boxed{\textsc{Variadic}} \qquad \frac{n : \mathsf{name} \quad p : \mathsf{type}}{n : p : \mathsf{type}}\ \boxed{\textsc{Named}}$$

Brackets:

$$\frac{p : \mathsf{type}}{[p] : \mathsf{type}}\ \boxed{\textsc{Brackets}}$$

Functions:

$$\frac{a : \mathsf{type} \quad b : \mathsf{type}}{a \rightarrow b : \mathsf{type}}\ \boxed{\textsc{Function}}$$

Figure 1: Typify syntax rules

level	name		associativity	example
8	optional	?	postfix	$a?$
7	application	$a\ b$	left associative	$a\ b\ c \equiv (a\ b)\ c$
6	conjunction	$\wedge$	associative	$(a \wedge b) \wedge c \equiv a \wedge (b \wedge c)$
5	disjunction	$\vee$	associative	$(a \vee b) \vee c \equiv a \vee (b \vee c)$
4	ellipsis	$\ldots$	postfix	$a \ldots$
3	type	:	right associative	$x : a$
2	product	$\times$	associative	$(a \times b) \times c \equiv a \times (b \times c)$
1	function	$\rightarrow$	right associative	$a \rightarrow b \rightarrow c \equiv a \rightarrow (b \rightarrow c)$
0	semicolon	;	associative	

Table 1: Operator precedence

name		code variants
any	$\top$	$\top$ *
nothing	$\bot$	$\bot$ _|_
unit	$\mathbb{1}$	$\mathbb{1}$ ()
optional	?	?
conjunction	$\wedge$	$\wedge$ |
disjunction	$\vee$	$\vee$ &
ellipsis	$\ldots$	$\ldots$ (either three dots, or unicode ellipsis)
type	:	:
product	$\times$	$\times$,
function	$\rightarrow$	$\rightarrow$ ->
semicolon	;	;

Table 2: Special character code representations

$$
\begin{aligned}
& a \times ys : b \ldots \times c \rightarrow d \rightarrow e \\
&\quad \equiv x : a \times ys : (b \ldots) \times c \rightarrow d \rightarrow e \\
&\quad \equiv (x : a) \times (ys : (b \ldots)) \times c \rightarrow d \rightarrow e \\
&\quad \equiv ((x : a) \times (ys : (b \ldots)) \times c) \rightarrow d \rightarrow e \\
&\quad \equiv ((x : a) \times (ys : (b \ldots)) \times c) \rightarrow (d \rightarrow e) \\
& a \times y : b \rightarrow c \\
&\quad \equiv a \times (y : b) \rightarrow c \\
&\quad \equiv (a \times (y : b)) \rightarrow c
\end{aligned}
$$

$$
\begin{aligned}
& a \vee b? \wedge c\ d \ldots \rightarrow e \\
&\quad \equiv a \vee (b?) \wedge (c\ d) \ldots \rightarrow e \\
&\quad \equiv a \vee ((b?) \wedge (c\ d)) \ldots \rightarrow e \\
&\quad \equiv (a \vee ((b?) \wedge (c\ d))) \ldots \rightarrow e \\
&\quad \equiv ((a \vee ((b?) \wedge (c\ d))) \ldots) \rightarrow e
\end{aligned}
$$

Figure 2: Examples of operator precedence

$$
\begin{aligned}
& \mathit{Either}\ a\ b \equiv \{\mathit{type} : \texttt{"left"};\ \mathit{value} : a\} \vee \{\mathit{type} : \texttt{"right"};\ \mathit{value} : b\} \\
& \mathit{flatMap} : (@ : \mathit{Observable}\ A \times f : A \rightarrow \mathit{Observable}\ B \vee \mathit{Event}\ B \vee B) \rightarrow \mathit{EventStream}\ B
\end{aligned}
$$

Figure 3: Examples of real world types