## Typify types

## Oleg Grenrus

November 21, 2014

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

## 1 Formal syntax definition

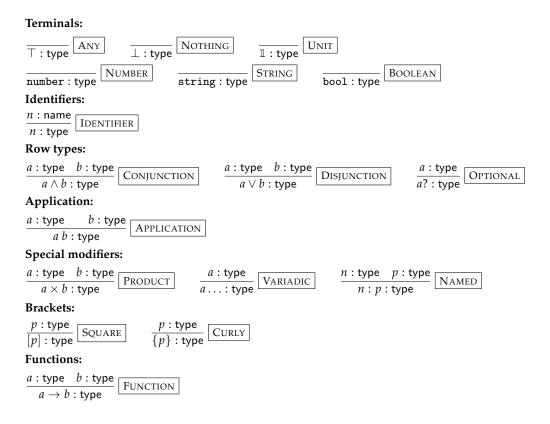


Figure 1: Typify syntax rules

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	$(a \lor b) \lor c \equiv a \lor (b \lor c)$
4	ellipsis		postfix	$a\dots$
3	type	:	left associative	$x:a:b\equiv(x:a):b$
2	product	X	associative	$(a \times b) \times c \equiv a \times (b \times c)$
1	function	$\rightarrow$	right associative	$a \to b \to c \equiv a \to (b \to c)$

Table 1: Operator precedence

name		code variants
any	Т	T *
nothing	$\perp$	⊥_ _
unit	1	1 ()
optional	?	?
conjunction	$\wedge$	$\wedge$
disjunction	$\vee$	∨ &
ellipsis		(either three dots, or unicode ellipsis)
type	:	:
product	×	× ,
function	$\rightarrow$	$\rightarrow$ ->

Table 2: Special character code representations

```
\begin{array}{l} a\times ys:b\ldots\times c\to d\to e\\ \equiv x:a\times ys:(b\ldots)\times c\to d\to e\\ \equiv (x:a)\times (ys:(b\ldots))\times c\to d\to e\\ \equiv ((x:a)\times (ys:(b\ldots))\times c\to d\to e\\ \equiv (a\vee ((b?)\wedge (c\ d)))\ldots\to e\\ \equiv (a\vee ((b?)\wedge (c\ d)))\ldots\to e\\ \equiv (a\vee ((b?)\wedge (c\ d)))\ldots\to e\\ \equiv (a\times (y:b)\to c\\ \equiv (a\times (y:b)\to c\\ \end{array}
```

Figure 2: Examples of operator precedence

```
Either a b \equiv \{type : "left", value : a\} \lor \{type : "right", value : b\}

flatMap : (@ : Observable \ A \times f : A \rightarrow Observable \ B \lor Event \ B \lor B) \rightarrow EventStream \ B
```

Figure 3: Examples of real world types