F-Safe, syntaxe concrète

Aurélien Deharbe

10 février 2012

Résumé

Ce document résume toutes les constructions syntaxiques du langage F-Safe.

Whole programs :

$$prog \ ::= \ tdef^*\ vdef^*\ a^*$$

Expressions :

a, b	$::=$	x	identifier
		$\mid C$	constant constructor
		$\mid C(a_1, \dots, a_n)$	constructor with arguments
		$\mid C[\tau_1, \dots, \tau_m]$	constant parameterized constructor
		$\mid C[\tau_1, \dots, \tau_m](a_1, \dots, a_n)$	parameterized constructor with arguments
		$\mid \{(a_1, b_1), \dots, (a_n, b_n)\}[\tau_1 \rightarrow \tau_2]$	applicative constructor
		$\mid (a)$	parenthesized expression
		$\mid \texttt{fun}\ (a_1 : \tau_1, \dots, a_n : \tau_n) : \tau \Rightarrow a$	function abstraction
		$\mid \texttt{fun}\ [tvar_1, \dots, tvar_m](a_1 : \tau_1, \dots, a_n : \tau_n) : \tau \Rightarrow a$	parameterized function abstraction
		$\mid a(a_1, \dots, a_n)$	function application
		$\mid a[\tau_1, \dots, \tau_m](a_1, \dots, a_n)$	parameterized function application
		$\mid \texttt{let}\ (x_1 : \tau_1 = a_1, \dots, x_n : \tau_n = a_n)\ \{\ a\ \}$	let binding
		$\mid \texttt{case}\ a_1, \dots, a_m\ \{\ \mid f_1 \Rightarrow b_1 \mid \dots \mid f_n \Rightarrow b_n\ \}$	pattern-matching

Patterns :

f	$::=$	$f_1, \dots, f_n$	multiple patterns
		$\mid x : \tau$	variable identifier
		$\mid \_ : \tau$	anonymous variable
		$\mid C$	constant constructor
		$\mid C(f_1, \dots, f_n)$	constructor with arguments
		$\mid C[\tau_1, \dots, \tau_m]$	parameterized constructor
		$\mid C[\tau_1, \dots, \tau_m](f_1, \dots, f_n)$	parameterized constructor with arguments
		$\mid \{\}[\tau_1 \rightarrow \tau_2]$	empty relation
		$\mid \{(x_1 : \tau_1, x_2 : \tau_2), x : \texttt{Map}[\tau_1 \rightarrow \tau_2]\}$	arbitrary selection in relation

Type expressions :

τ	$::=$	$tvar$	type variable
		$\mid \tau_1 \rightarrow \tau_2$	function type
		$\mid tname$	type constructor
		$\mid tname[\tau_1, \dots, \tau_n]$	parameterized type constructor
		$\mid \texttt{Map}[\tau_1 \rightarrow \tau_2]$	applicative type
		$\mid (\tau)$	parenthesized type

Type definitions :

$tdef$ ::=	`type` $tname = cstr_1 \mid \ldots \mid cstr_n$	simple type
	$\mid$ `type` $tname[tvar_1, \ldots, tvar_m] = cstr_1 \mid \ldots \mid cstr_n$	parameterized type
	$\mid tdef_1$ `and` $tdef_2$	mutually inductive types

Constructor definitions :

$cstr$ ::=	C	constant constructor
	$\mid C(x_1 : \tau_1, \ldots, x_n : \tau_n)$	constructor with arguments

Global variables definitions :

$vdef$ ::=	`def` $x_1 : \tau_1, \ldots, x_n : \tau_n = a_1, \ldots, a_n$	standard definition
	$\mid$ `def` $x(x_1 : \tau_1, \ldots, x_n : \tau_n) : \tau = a$	syntactic shortcut for function definition