LambdaScript Syntax and Semantics

Alex Kozik

July 15, 2023

Syntax

Metavariables

Below is a list of meta-variables for different fundamental langauge constructs

$x \in Var$	Variable indentifier
$b \in \{true, false\}$	Boolean
$n \in \mathbb{N}$	Natural number
$s \in \Sigma^*$	String
$\oplus \in \{+, -, *, /, \%, <, >, <=, >=, ==, != \}$	Binary operator
	Unary operator

Expressions

$\langle e \rangle ::=$	n	Integer
$\mid$	b	Boolean
$\mid$	s	String
$\mid$	()	Nothing
$\mid$	x	Identifier
$\mid$	$e_1 \oplus e_2$	Binary Operation
$\mid$	$(e_1, e_2, ..., e_n)$	Vector
$\mid$	$[\,]$	Nil (empty list)
$\mid$	$e_1 :: e_2$	Cons (nonempty list)
$\mid$	**fn** $p \rightarrow e$	Function
$\mid$	**bind** p $\leftarrow e_1$ **in** e_2	Bind expression
$\mid$	**bind** $p \ \ p_1 \ ... \ p_n \leftarrow e_1$ **in** e_2	Bind expression
$\mid$	**bind rec** $f \leftarrow$ **fn** p $\rightarrow e_1$ **in** e_2	Recursive function bind
$\mid$	**bind rec** $f \ \ p_1 \ ... \ p_n \leftarrow e_1$ in e_2	Recursive function bind
$\mid$	$e_1 \ \ e_2$	Function application
$\mid$	**if** e_1 **then** e_2 **else** e_3	Ternary expressions
$\mid$	**switch** $e_0 => \mid p_1 \rightarrow e_1 \ ... \ \mid p_n \rightarrow e_n$ **end**	Switch expression

Patterns

$\langle p \rangle ::=$	_	Wildcard pattern
$\mid$	x	Identifier pattern
$\mid$	()	Nothing pattern
$\mid$	b	Boolean pattern
$\mid$	n	Integer pattern
$\mid$	s	String pattern
$\mid$	$(p_1, p_2, ..., p_n)$	Vector pattern
$\mid$	$[\,]$	Nil pattern
$\mid$	$p_1 :: p_2$	Cons pattern

Values

$\langle v \rangle$ ::=	n	Integer value
	s	String value
	b	Boolean value
	()	Nothing value
	[ ]	Nil value
	$v_1 :: v_2$	Cons value
	$(\Delta,\ p,\ e)$	Function Closure