

$t ::=$		terms
	$b$	boolean value
	$n$	numeric value
	$op$	operator
	$(\lambda x. t)^{T \rightarrow T}$	abstraction
	$x$	variable
	$t t$	application
	$\text{mlet } x : T = t \text{ in } t$	overloading let
	$t :: T$	ascription
$b ::=$		boolean value
	$\text{true}$	true value
	$\text{false}$	false value
$op ::=$		operators
	$\text{add1}$	sum
	$\text{not}$	negation
$T ::=$		types
	$\text{Int}$	type of integers
	$\text{Bool}$	type of booleans
	$T \rightarrow T$	type of functions
$v ::=$		configuration – values
	$b$	boolean value
	$n$	numeric value
	$op$	operator
	$((\lambda x. t)^{T \rightarrow T})[s]$	closure
$c ::=$		configurations
	$v$	
	$t[s]$	
	$c c$	
	$\text{mlet } x : T = c \text{ in } c$	
	$c :: T$	
	$\text{error}$	
$s ::=$		explicit substitutions
	$\bullet$	empty substitution
	$x \mapsto \{(\overline{v : T})\}, s$	variable substitution

Figure 1: Syntax of the simply typed lambda-calculus with overloading.

- Deterministic.
- Type error detection.
- Dispatch error detection.
- Ambiguity error detection.
- Type annotation in lambda functions, `mlet` and ascription.
- More expressive than Semantic 3, with the use structural tags.
- Do not support context-dependent overloading.

**Definition 1** ( $\oplus$ ). *Given an environment  $s$  and a variable binding  $x \mapsto (v_1 : T_1)$ , the operator  $\oplus$  is defined as follows:*

$$s \oplus x \mapsto (v_1 : T_1) = \begin{cases} x \mapsto \{(v_1 : T_1)\} & s = \emptyset \\ x \mapsto \{(\overline{v : T})\} \cup \{(v_1 : T_1)\}, s' & s = x \mapsto \{(\overline{v : T})\}, s' \\ y \mapsto \{(\overline{v : T})\}, s' \oplus x \mapsto (v_1 : T_1) & s = y \mapsto \{(\overline{v : S})\}, s' \end{cases}$$

	$c \longrightarrow c$
$b[s] \longrightarrow b$	(Bool)
$n[s] \longrightarrow n$	(Num)
$op[s] \longrightarrow op$	(Op)
$(t :: T)[s] \longrightarrow t[s] :: T$	(AscSub)
$(\text{mlet } x : T_1 = t_1 \text{ in } t_2)[s] \longrightarrow \text{mlet } x : T_1 = t_1[s] \text{ in } t_2[s]$	(LetSub)
$(t_1 \ t_2)[s] \longrightarrow t_1[s] \ t_2[s]$	(AppSub)
$v :: T \longrightarrow v$	(Asc)
$\text{mlet } x : T_1 = v \text{ in } t_2[s] \longrightarrow t_2[x \mapsto (v : T_1) \oplus s]$	(Let)
$((\lambda x. t_2)^{T_1 \rightarrow T_2})[s] \ v \longrightarrow ([x \mapsto v]t_2)[s]$	(App)
$\text{add1 } n \longrightarrow n + 1$	(Sum)
$\text{not } b \longrightarrow \neg b$	(Negation)

Figure 2: Configuration reduction rules.

**Definition 2** (flat). *The function flat is defined as follows:*

$$\text{flat}(s) = \begin{cases} \emptyset & s = \emptyset \\ x \mapsto (v_1 : T_1) \cdots, x \mapsto (v_n : T_n), \text{flat}(s') & s = x \mapsto \{(\overline{v : T})\}, s' \end{cases}$$

**Definition 3** (lookup). *The function lookup is defined as follows:*

$$\text{lookup}(x, s, S') = \begin{cases} v_i & s = x \mapsto \{(\overline{v : S})\}, s' \wedge S' = S_i \\ \text{lookup}(x, s', S') & s = y \mapsto \{(\overline{v : S})\}, s' \\ \text{error} & s = \emptyset \end{cases}$$

$$\text{lookup}(x_1, x_2, s_1, s_2) = \begin{cases} (v_1, v_2) & !\exists x_1 \mapsto (v_1 : T_1) \in \text{flat}(s_1) \wedge !\exists x_2 \mapsto (v_2 : T_1) \in \text{flat}(s_2) \\ \text{error} & \text{otrw} \end{cases}$$

**Definition 4** (tag). *The function tag is defined as follows:*

$$\text{tag}(v) = \begin{cases} \text{Int} & v = n \\ \text{Bool} & v = b \\ T_1 \rightarrow T_2 & v = (\lambda x. t_2)^{T_1 \rightarrow T_2} \end{cases}$$

	$c \longrightarrow c$
$\frac{v = \text{lookup}(x, [s], T)}{x[s] :: T \longrightarrow v :: T}$	(AscVar)
$\frac{v = \text{lookup}(x_1, [s_1], T_1)}{\text{mlet } x : T_1 = x_1[s_1] \text{ in } c_2 \longrightarrow \text{mlet } x : T_1 = v \text{ in } c_2}$	(LetVar)
$\frac{v_2 = \text{lookup}(x_2, [s_2], T_1)}{((\lambda x. t_2)^{T_1 \rightarrow T_2})[s] x_2[s_2] \longrightarrow ((\lambda x. t_2)^{T_1 \rightarrow T_2})[s] v_2}$	(AppVar1)
$\frac{T = \text{tag}(v_2) \quad v_1 = \text{lookup}(x_1, [s_1], T \rightarrow *)}{x_1[s_1] v_2 \longrightarrow v_1 v_2}$	(AppVar2)
$\frac{(v_1, v_2) = \text{lookup}(x_1, x_2, [s_1], [s_2])}{x_1[s_1] x_2[s_2] \longrightarrow v_1 v_2}$	(AppVar3)
$\frac{n = \text{lookup}(x, [s], \text{Int})}{\text{add1 } x[s] \longrightarrow \text{add1 } n}$	(SumVar)
$\frac{b = \text{lookup}(x, [s], \text{Bool})}{\text{not } x[s] \longrightarrow \text{not } b}$	(NegationVar)
$\frac{c \longrightarrow c' \quad \text{notVal\_Var}(c)}{c :: T \longrightarrow c' :: T}$	(Asc1)
$\frac{c_1 \longrightarrow c'_1 \quad \text{notVal\_Var}(c_1)}{\text{mlet } x : T_1 = c_1 \text{ in } c_2 \longrightarrow \text{mlet } x : T_1 = c'_1 \text{ in } c_2}$	(Let1)
$\frac{c_1 \longrightarrow c'_1 \quad \text{notVal\_Var}(c_1)}{c_1 c_2 \longrightarrow c'_1 c_2}$	(App1)
$\frac{c \longrightarrow c' \quad \text{notVal\_Var}(c)}{v c \longrightarrow v c'}$	(App2)
$\frac{c_2 \longrightarrow c'_2 \quad \text{notVal\_Var}(c_2)}{x[s] c_2 \longrightarrow x[s] c'_2}$	(App3)

Figure 3: Configuration reduction rules.