t	$::=$		terms
		b	boolean value
		n	numeric value
		op	operator
		$(\lambda x.\ t)^{T \to T}$	abstraction
		x	variable
		$t\ t$	application
		$\mathsf{mlet}\ x : T = t\ \mathsf{in}\ t$	overloading let
		$t :: T$	ascription
b	$::=$		boolean value
		true	true value
		false	false value
op	$::=$		operators
		add1	sum
		not	negation
T	$::=$		types
		Int	type of integers
		Bool	type of booleans
		$T \to T$	type of functions

v	$::=$		configuration − values
		b	boolean value
		n	numeric value
		op	operator
		$((\lambda x.\ t)^{T \to T})[s]$	closure
c	$::=$		configurations
		v	
		$t[s]$	
		$c\ c$	
		$\mathsf{mlet}\ x : T = c\ \mathsf{in}\ c$	
		$c :: T$	
		error	
s	$::=$		explicit substitutions
		$\bullet$	empty substitution
		$x \mapsto \{(\overline{v : T})\}, s$	variable substitution

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

$$\boxed{c \longrightarrow c}$$

$$b[s] \longrightarrow b \qquad \textsf{(Bool)}$$

$$n[s] \longrightarrow n \qquad \textsf{(Num)}$$

$$op[s] \longrightarrow op \qquad \textsf{(Op)}$$

$$(t :: T)[s] \longrightarrow t[s] :: T \qquad \textsf{(AscSub)}$$

$$(\textsf{mlet } x : T_1 = t_1 \textsf{ in } t_2)[s] \longrightarrow \textsf{mlet } x : T_1 = t_1[s] \textsf{ in } t_2[s] \qquad \textsf{(LetSub)}$$

$$(t_1\ t_2)[s] \longrightarrow t_1[s]\ t_2[s] \qquad \textsf{(AppSub)}$$

$$v :: T \longrightarrow v \qquad \textsf{(Asc)}$$

$$\textsf{mlet } x : T_1 = v \textsf{ in } t_2[s] \longrightarrow t_2[x \mapsto (v : T_1) \oplus s] \qquad \textsf{(Let)}$$

$$((\lambda x.\ t_2)^{T_1 \to T_2})[s]\ v \longrightarrow ([x \mapsto v]t_2)[s] \qquad \textsf{(App)}$$

$$\textsf{add1 } n \longrightarrow n + 1 \qquad \textsf{(Sum)}$$

$$\textsf{not } b \longrightarrow \neg\ b \qquad \textsf{(Negation)}$$

Figure 2: Configuration reduction rules.

$$\boxed{c \longrightarrow c}$$

$$\frac{v = \mathsf{lookup}(x, [s], T)}{x[s] :: T \longrightarrow v :: T} \quad \text{(AscVar)}$$

$$\frac{v = \mathsf{lookup}(x_1, [s_1], T_1)}{\mathsf{mlet}\ x : T_1 = x_1[s_1]\ \mathsf{in}\ c_2 \longrightarrow \mathsf{mlet}\ x : T_1 = v\ \mathsf{in}\ c_2} \quad \text{(LetVar)}$$

$$\frac{v_2 = \mathsf{lookup}(x_2, [s_2], T_1)}{((\lambda x.\ t_2)^{T_1 \to T_2})[s]\ x_2[s_2] \longrightarrow ((\lambda x.\ t_2)^{T_1 \to T_2})[s]\ v_2} \quad \text{(AppVar1)}$$

$$\frac{T = \mathsf{tag}(v_2) \qquad v_1 = \mathsf{lookup}(x_1, [s_1], T \to *)}{x_1[s_1]\ v_2 \longrightarrow v_1\ v_2} \quad \text{(AppVar2)}$$

$$\frac{(v_1, v_2) = \mathsf{lookup}(x_1, x_2, [s_1], [s_2])}{x_1[s_1]\ x_2[s_2] \longrightarrow v_1\ v_2} \quad \text{(AppVar3)}$$

$$\frac{n = \mathsf{lookup}(x, [s], \mathsf{Int})}{\mathsf{add1}\ x[s] \longrightarrow \mathsf{add1}\ n} \quad \text{(SumVar)}$$

$$\frac{b = \mathsf{lookup}(x, [s], \mathsf{Bool})}{\mathsf{not}\ x[s] \longrightarrow \mathsf{not}\ \ b} \quad \text{(NegationVar)}$$

$$\frac{c \longrightarrow c' \qquad \mathsf{notVal\_Var}(c)}{c :: T \longrightarrow c' :: T} \quad \text{(Asc1)}$$

$$\frac{c_1 \longrightarrow c_1' \qquad \mathsf{notVal\_Var}(c_1)}{\mathsf{mlet}\ x : T_1 = c_1\ \mathsf{in}\ c_2 \longrightarrow \mathsf{mlet}\ x : T_1 = c_1'\ \mathsf{in}\ c_2} \quad \text{(Let1)}$$

$$\frac{c_1 \longrightarrow c_1' \qquad \mathsf{notVal\_Var}(c_1)}{c_1\ c_2 \longrightarrow c_1'\ c_2} \quad \text{(App1)}$$

$$\frac{c \longrightarrow c' \qquad \mathsf{notVal\_Var}(c)}{v\ c \longrightarrow v\ c'} \quad \text{(App2)}$$

$$\frac{c_2 \longrightarrow c_2' \qquad \mathsf{notVal\_Var}(c_2)}{x[s]\ c_2 \longrightarrow x[s]\ c_2'} \quad \text{(App3)}$$

Figure 3: Configuration reduction rules.