t	$::=$		terms
		b	boolean value
		n	numeric value
		op	operator
		$\lambda x.\ t$	abstraction
		x	variable
		$t\ t$	application
		$\mathsf{mlet}\ x = t\ \mathsf{in}\ t$	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
		$T \rightarrow T$	type of functions

S	$::=$		tags
		Int	integer tag
		Bool	boolean tag
		Fun	function tag
v	$::=$		configuration – values
		b	boolean value
		n	numeric value
		op	operator
		$(\lambda x.\ t)[s]$	closure
c	$::=$		configurations
		v	
		$t[s]$	
		$c\ c$	
		$\mathsf{mlet}\ x = c\ \mathsf{in}\ c$	
		$c :: T$	
		error	
s	$::=$		explicit substitutions
		$\bullet$	empty substitution
		$x \mapsto \{(\overline{v : S})\}, s$	variable substitution

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

- Deterministic.
- Type error detection.
- Dispatch error detection. In the case of the lambda functions, it is not effective because if the environment contains at least a value with tag function, it is not detected dispatch error, because we don't have type information for lambdas function. Similar with mlet, expression.
- Ambiguity error detection(restrictive).
- Without type annotation in lambda functions or mlet, only in ascription.
- Semantic "tag driven", introducing flat tag in the environment.

Definition 1 ($\oplus$). *Given an environment s and a variable binding $x \mapsto (v_1 : S_1)$, the operator $\oplus$ is defined as follows:*

$$s \oplus x \mapsto (v_1 : S_1) = \begin{cases} x \mapsto \{(v_1 : S_1)\} & s = \varnothing \\ x \mapsto \{(\overline{v : S})\} \cup \{(v_1 : S_1)\}, s' & s = x \mapsto \{(\overline{v : S})\}, s' \\ y \mapsto \{(\overline{v : S})\}, s' \oplus x \mapsto (v_1 : S_1) & s = y \mapsto \{(\overline{v : S})\}, s' \end{cases}$$

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

$$b[s] \longrightarrow b \qquad \text{(False)}$$

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

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

$$x[\,] \longrightarrow \mathsf{error} \qquad \text{(ErrVarFail)}$$

$$x[x \mapsto \{(\overline{v : S_1})\}, s' \longrightarrow v_i \qquad \text{(VarOk)}$$

$$\frac{x \neq y}{x[y \mapsto \{(\overline{v : S_1})\}, s] \longrightarrow x[s]} \qquad \text{(VarNext)}$$

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

$$\frac{S_1 = \mathsf{tagVal}(v)}{\mathsf{mlet}\ x = v\ \mathsf{in}\ t_2[s] \longrightarrow t_2[x \mapsto (v : S_1) \oplus s]} \qquad \text{(Let)}$$

$$(\lambda x.\ t_2)[s]\ v \longrightarrow ([x \mapsto v]t_2)[s] \qquad \text{(App)}$$

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

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

Figure 2: Configuration reduction rules.

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

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

$$\frac{v_1 = \mathsf{lookup}(x_1, [s_1], \mathsf{Fun})}{x_1[s_1]\ v_2 \longrightarrow v_1\ v_2} \quad (\mathsf{AppVar})$$

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

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

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

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

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

$$\frac{c \longrightarrow c' \qquad \mathsf{notVal}(c)}{(\lambda x.\ t_2)[s]\ c \longrightarrow (\lambda x.\ t_2)[s]\ c'} \quad (\mathsf{App2})$$

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

$$\frac{c \longrightarrow c' \qquad \mathsf{notVal\_Var}(c)}{op\ c \longrightarrow op\ c'} \quad (\mathsf{App4})$$

Figure 3: Configuration reduction rules.

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

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

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

$$\mathsf{lookup}(x, s, S') = \begin{cases} v_i & !\exists(x \mapsto (v_i : S)) \in \mathsf{flat}(s) \\ \mathsf{lookup}(x, s', S') & s = y \mapsto \{\overline{(v : S)}\}, s' \\ \mathsf{error} & s = \varnothing \end{cases}$$

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

$$\mathsf{tagType}(T) = \begin{cases} \mathsf{Int} & T = \mathsf{Int} \\ \mathsf{Bool} & T = \mathsf{Bool} \\ \mathsf{Fun} & T = T_1 \to T_2 \end{cases}$$

Definition 5 (tagVal). *The function* tagVal *is defined as follows:*

$$\mathsf{tagVal}(v) = \begin{cases} \mathsf{Int} & v = n \\ \mathsf{Bool} & v = b \\ \mathsf{Fun} & v = \lambda x.\ t \end{cases}$$