

$t$	$::=$	$x$ $\lambda x : T. t$ $t_1 t_2$	terms variable abstraction application
$c$	$::=$	$t[s]$ $c c$	configurations
$v$	$::=$	$(\lambda x : T. t)[s]$	values closure
$T$	$::=$	$T \rightarrow T$	types type of functions
$\Gamma$	$::=$	$\emptyset$ $\Gamma, x : T$	contexts empty context term variable binding
$s$	$::=$	$\bullet$ $(x, v) : s$	explicit substitutions empty substitution variable substitution

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

**Lemma 1** (Inversion of term typing).

1. If  $\Gamma \vdash x : R$ , then  $x : R \in \Gamma$
2. If  $\Gamma \vdash \lambda x : T_1. t_2 : R$ , then  $R = T_1 \rightarrow R_2$  for some  $R_2$ , with  $\Gamma, x : T_1 \vdash t_2 : R_2$ .
3. If  $\Gamma \vdash t_1 t_2 : R$ , then there is some type  $T_{11}$  such that  $\Gamma \vdash t_1 : T_{11} \rightarrow R$  and  $\Gamma \vdash t_2 : T_{11}$ .

*Proof.* Immediate from the definition of the typing relation.  $\square$

**Lemma 2** (Inversion of configuration typing).

1. If  $\vdash_c x[s] : R$ , then  $(x, v) \in s$ , for some  $v$ , and  $\vdash_c v : R$ .
2. If  $\vdash_c (\lambda x : T_1. t_2)[s] : R$ , then  $R = T_1 \rightarrow R_2$  for some  $R_2$ , with  $\Gamma(s), x : T_1 \vdash t_2 : R_2$ .
3. If  $\vdash_c (t_1 t_2)[s] : R$ , then  $\vdash_c t_1[s] t_2[s] : R$ .
4. If  $\vdash_c c_1 c_2 : R$ , then there is some type  $T_{11}$  such that  $\vdash_c c_1 : T_{11} \rightarrow R$  and  $\vdash_c c_2 : T_{11}$ .

*Proof.* Immediate from the definition of the typing relation.  $\square$

$\frac{x : T \in \Gamma}{\Gamma \vdash x : T}$	$\boxed{\Gamma \vdash t : T}$ (TVar)	$\frac{(x, v) \in s \quad \vdash_c v : T}{\vdash_c x[s] : T}$	$\boxed{\vdash_c c : T}$ (TCVar)
$\frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1. t_2 : T_1 \rightarrow T_2}$	(TAbs)	$\frac{\Gamma(s), x : T_1 \vdash t_2 : T_2}{\vdash_c (\lambda x : T_1. t_2)[s] : T_1 \rightarrow T_2}$	(TCAbs)
$\frac{\Gamma \vdash t_1 : T_{11} \rightarrow T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1 t_2 : T_{12}}$	(TApp)	$\frac{\vdash_c t_1[s] t_2[s] : T_{12}}{\vdash_c (t_1 t_2)[s] : T_{12}}$	(TCApp)
		$\frac{\vdash_c c_1 : T_{11} \rightarrow T_{12} \quad \vdash_c c_2 : T_{11}}{\vdash_c c_1 c_2 : T_{12}}$	(TCCApp)

Figure 2: Term and configuration typing rules.

	$\boxed{c \longrightarrow c}$
$x[(x, v) : s] \longrightarrow v$	(VarOk)
$\frac{x \neq y}{x[(y, v) : s] \longrightarrow x[s]}$	(VarNext)
$(t_1 t_2)[s] \longrightarrow t_1[s] t_2[s]$	(AppSub)
$(\lambda x : T_1. t_2)[s] v \longrightarrow t_2[(x, v) : s]$	(App)
$\frac{c_1 \longrightarrow c'_1}{c_1 c_2 \longrightarrow c'_1 c_2}$	(App1)
$\frac{c \longrightarrow c'}{v c \longrightarrow v c'}$	(App2)

Figure 3: Configuration reduction rules.

**Lemma 3** (Canonical Forms).

1. If  $v$  is a value of type  $T_1 \rightarrow T_2$ , then  $v = (\lambda x : T_1. t_2)[s]$ .

*Proof.* Straightforward.  $\square$

**Definition 1** ( $\Gamma(s)$ ). The typing context built from a substitution  $s$ , writing  $\Gamma(s)$ , it is defined as follows:

$$\Gamma(s) = \begin{cases} \emptyset & s = \bullet \\ \Gamma(s'), x : T & s = (x, v) : s' \wedge \vdash_c v : T \end{cases}$$

**Theorem 4** (Progress). Suppose  $c$  is a well-typed configuration (that is,  $\vdash_c c : T$  for some  $T$ ). Then either  $c$  is a value or else there is some  $c'$  such that  $c \longrightarrow c'$ .

*Proof.* By induction on a derivation of  $\vdash_c c : T$ .

*Case* (TCVar). Then  $c = x[s]$ , with  $(x, v) \in s$ , for some  $v$ , and  $\vdash_c v : T$ . Since  $x \in \text{dom}(s)$ , if the substitution  $s = (x, v) : s'$ , then rule **VarOk**, applies, otherwise, rule **VarNext** applies.

*Case* (TCAbs). Then  $c = (\lambda x : T_1. t_2)[s]$ . This case is immediate, since closures are values.

*Case* (TCApp). Then  $c = (t_1 t_2)[s]$ , so rule **AppSub** applies to  $c$ .

*Case* (TCCApp). Then  $c = c_1 c_2$ , with  $\vdash_c c_1 : T_{11} \rightarrow T$ , for some  $T_{11}$  and  $\vdash_c c_2 : T_{11}$ , by the Lemma 2. Then, by the induction hypothesis, either  $c_1$  is a value or else it can take a step of evaluation, and likewise  $c_2$ . If  $c_1$  can take a step, then rule **App1** applies to  $c$ . If  $c_1$  is a value and  $c_2$  can take a step, then rule **App2** applies. Finally, if both  $c_1$  and  $c_2$  are values, then the Lemma 3 tells us that  $c_1$  has the form  $(\lambda x : T_{11}. t_{12})[s]$ , and so rule **App** applies to  $c$ .  $\square$

**Definition 2** (Well typed substitution). A substitution  $s$  is said well typed with a typing context  $\Gamma$ , writing  $\Gamma \vdash s$ , if  $\text{dom}(s) = \text{dom}(\Gamma)$  and for every  $(x, v) \in s$  and  $\vdash_c v : T$ , where  $x : T \in \Gamma$ .

**Lemma 5** (Permutation). If  $\Gamma \vdash t : T$  and  $\Delta$  is a permutation of  $\Gamma$ , then  $\Delta \vdash t : T$ .

*Proof.* By induction on typing derivations.  $\square$

**Lemma 6.** If  $\Gamma \vdash s$  then  $\Gamma$  is a permutation of  $\Gamma(s)$ .

*Proof.* By the definition of well typed substitution.  $\square$

**Lemma 7.** If  $\Gamma \vdash s$  and  $\vdash_c v : T$ , then  $\Gamma, x : T \vdash (x, v) : s$ .

*Proof.* By the definition of well typed substitution.  $\square$

**Lemma 8.** If  $\Gamma \vdash s$  then  $\vdash_c t[s] : T$  if and only if  $\Gamma \vdash t : T$ .

*Proof.* By induction on typing derivations, using Lemma 5 and Lemma 6.  $\square$

**Theorem 9** (Preservation). *If  $\vdash_c c : T$  and  $c \longrightarrow c'$ , then  $\vdash_c c' : T$ .*

*Proof.* By induction on a derivation of  $\vdash_c c : T$ .

*Case* (TCVar). Then  $c = x[s]$ , with  $\vdash_c (x, v) \in s$ , for some  $v$ , and  $\vdash_c v : T$ . We find that there are two rule by which  $c \longrightarrow c'$  can be derived: *VarOk* and *VarNext*. We consider each case separately.

- *Subcase* (VarOk). Then  $s = (x, v) : s'$  and  $c' = v$ . Since  $(x, v) \in s$  and  $\vdash_c v : T$ , then  $\vdash_c c' : T$ .
- *Subcase* (VarNext). Then  $s = (y, v) : s'$ ,  $x \neq y$  and  $c' = x[s']$ . Since  $(x, v) \in s'$  too, and  $\vdash_c v : T$  then  $\vdash_c x[s'] : T$ , that is  $\vdash_c c' : T$ .

*Case* (TAbs). Then  $c = (\lambda x : T_1. t_2)[s]$ . It cannot be the case that  $c \longrightarrow c'$ , because  $c$  is a value, then the requirements of the theorem are vacuously satisfied.

*Case* (TCApp). Then  $c = (t_1 t_2)[s]$  and  $\vdash_c t_1[s] t_2[s] : T$ . We find that there are only one rule by which  $c \longrightarrow c'$  can be derived: *AppSub*. With this rule  $c' = t_1[s] t_2[s]$ , then we can conclude that  $\vdash_c c' : T$ .

*Case* (TCCApp). Then  $c = c_1 c_2$ ,  $\vdash_c c_1 : T_2 \rightarrow T$  and  $\vdash_c c_2 : T_2$ . We find that there are three rules by which  $c \longrightarrow c'$  can be derived: *App1*, *App2* and *App*. We consider each case separately.

- *Subcase* (App1). Then  $c_1 \longrightarrow c'_1$ ,  $c' = c'_1 c_2$ . By the induction hypothesis,  $\vdash_c c'_1 : T_2 \rightarrow T$ , then we can apply rule *TCCApp*, to conclude that  $\vdash_c c'_1 c_2 : T$ , that is  $\vdash_c c' : T$ .
- *Subcase* (App2). Then  $c_2 \longrightarrow c'_2$ ,  $c' = c_1 c'_2$ . By the induction hypothesis,  $\vdash_c c'_2 : T_2$ , then we can apply rule *TCCApp*, to conclude that  $\vdash_c c_1 c'_2 : T$ , that is  $\vdash_c c' : T$ .
- *Subcase* (App): Then  $c_1 = (\lambda x : T_1. t_{12})[s]$ ,  $c_2 = v$ ,  $c' = t_{12}[(x, v) : s]$  and  $\Gamma(s), x : T_1 \vdash t_{12} : T$  by the Lemma 2. Since we know that  $\Gamma(s), x : T_1 \vdash (x, v) : s$  by the Lemma 7, the resulting configuration  $\vdash_c t_{12}[(x, v) : s] : T$ , by the Lemma 8, that is  $\vdash_c c' : T$ .

$\square$