

Chapter 3

Simple Types

Definition 3.0.7 (Redexes) *The redexes in \mathcal{L}_0 and their reducts are defined below.*

- $cf(\vec{v})$ is a redex, and each defined value of $cf(\vec{v})$ is a reduct of $cf(\vec{v})$.
- $\mathbf{if}(true, e_1, e_2)$ is a redex, and its reduct is e_1 .
- $\mathbf{if}(false, e_1, e_2)$ is a redex, and its reduct is e_2 .
- $\mathbf{fst}(\langle v_1, v_2 \rangle)$ is a redex, and its reduct is v_1 .
- $\mathbf{snd}(\langle v_1, v_2 \rangle)$ is a redex, and its reduct is v_2 .
- $(\lambda x.e)v$ is a redex, and its reduct is $e[x := v]$.
- $\mathbf{fix} f.e$ is a redex, and its reduct is $e[f := \mathbf{fix} f.e]$.

For instance, if we assume that $+$ represents the usual addition function on integers, then $1 + 2$ is a redex and 3 is the only reduct of $1 + 2$. More interestingly, we may also assume the existence of a nullary constant function $random$ such that $random()$ is a redex and every natural number is a reduct of $random()$.

Definition 3.0.8 (Evaluation) *We use \rightarrow for the binary evaluation relation on expressions. Given e_1 and e_2 , $e_1 \rightarrow e_2$ holds if $e_1 = E[e]$ and $e_2 = E[e']$ for some evaluation context E , redex e and a reduct e' of e . We use \rightarrow^* for the reflexive and transitive closure of \rightarrow .*

| | |
|---------------------|--|
| expressions | $e ::= x \mid c(\vec{e}) \mid \mathbf{if}(e_0, e_1, e_2) \mid \langle e_1, e_2 \rangle \mid \mathbf{fst}(e) \mid \mathbf{snd}(e) \mid \lambda x.e \mid (e_1)e_2 \mid \mathbf{fix} f.e$ |
| values | $v ::= x \mid cc(\vec{v}) \mid \langle v_1, v_2 \rangle \mid \lambda x.e$ |
| types | $T ::= \delta \mid T_1 * T_2 \mid T_1 \rightarrow T_2$ |
| typing contexts | $\Gamma ::= \emptyset \mid \Gamma, x : T$ |
| evaluation contexts | $E ::= [] \mid c(\vec{v}, E, \vec{e}) \mid \mathbf{if}(E, e_1, e_2) \mid \langle E, e \rangle \mid \langle v, E \rangle \mid \mathbf{fst}(E) \mid \mathbf{snd}(E) \mid (E)e \mid (v)E$ |

Figure 3.1: The syntax for \mathcal{L}_0

$$\begin{array}{c}
\frac{\Gamma(x) = T}{\Gamma \vdash x : T} \text{ (ty-var)} \\
\frac{\vdash c : (T_1, \dots, T_n) \rightarrow T \quad \Gamma \vdash e_i : T_i \text{ for } 1 \leq i \leq n}{\Gamma \vdash c(e_1, \dots, e_n) : T} \text{ (ty-cst)} \\
\frac{\Gamma \vdash e_0 : \text{bool} \quad \Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T}{\Gamma \vdash \text{if}(e_0, e_1, e_2) : T} \text{ (ty-if)} \\
\frac{\Gamma \vdash e_1 : T_1 \quad \Gamma \vdash e_2 : T_2}{\Gamma \vdash \langle e_1, e_2 \rangle : T_1 * T_2} \text{ (ty-tup)} \\
\frac{\Gamma \vdash e : T_1 * T_2}{\Gamma \vdash \text{fst}(e) : T_1} \text{ (ty-fst)} \\
\frac{\Gamma \vdash e : T_1 * T_2}{\Gamma \vdash \text{snd}(e) : T_2} \text{ (ty-snd)} \\
\frac{\Gamma, x : T_1 \vdash e : T_2}{\Gamma \vdash \lambda x. e : T_1 \rightarrow T_2} \text{ (ty-lam)} \\
\frac{\Gamma \vdash e_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash e_2 : T_1}{\Gamma \vdash (e_1)e_2 : T_2} \text{ (ty-app)} \\
\frac{\Gamma, f : T \vdash e : T}{\Gamma \vdash \text{fix } f. e : T} \text{ (ty-fix)}
\end{array}$$

Figure 3.2: The typing rules for expressions in \mathcal{L}_0

A typing judgment in \mathcal{L}_0 is of the form $\Gamma \vdash e : T$.

Lemma 3.0.9 (Canonical Forms) *Assume that $\emptyset \vdash v : T$ is derivable.*

- *If $T = T_1 * T_2$ for some types T_1 and T_2 , then v is of the form $\langle v_1, v_2 \rangle$.*
- *If $T = T_1 \rightarrow T_2$ for some types T_1 and T_2 , then v is of the form $\lambda x.v_0$*
- *If $T = \delta$ for some base type δ , then v is of the form $cc(\vec{v})$, where cc is a constructor associated with δ .*

Proof By a inspection of the typing rules in Figure 3.3. ■

Lemma 3.0.10 (Substitution) *Assume that $\Gamma_0, \Gamma \vdash e : T$ is derivable and $\Gamma_0 \vdash \theta : \Gamma$ holds. Then $\Gamma_0 \vdash e[\theta] : T$ is also derivable.*

Proof We proceed by structural induction on the typing derivation \mathcal{D} of $\Gamma_0, \Gamma \vdash e : T$.

- *e is some variable x . Then $x \in \mathbf{dom}(\Gamma_0, \Gamma)$. If $x \in \mathbf{dom}(\Gamma)$, then $\Gamma_0 \vdash \theta : \Gamma$ implies that $\Gamma_0 \vdash e[\theta] : \Gamma(x)$ is derivable since $e[\theta] = \theta(x)$. If $x \notin \mathbf{dom}(\Gamma)$, then $x \in \mathbf{dom}(\Gamma_0)$ and thus $\Gamma_0 \vdash x : T$ is derivable.*
- *e is of the form $\langle e_1, e_2 \rangle$. Then \mathcal{D} must be of the following form:*

$$\frac{\mathcal{D}_1 :: \Gamma_0, \Gamma \vdash e_1 : T_1 \quad \mathcal{D}_2 :: \Gamma_0, \Gamma \vdash e_2 : T_2}{\Gamma_0, \Gamma \vdash e : T}$$

where $T = T_1 * T_2$. By induction hypotheses on \mathcal{D}_1 and \mathcal{D}_2 , both $\Gamma_0 \vdash e_1[\theta] : T_1$ and $\Gamma_0 \vdash e_2[\theta] : T_2$ are derivable. Hence, $\Gamma_0 \vdash \langle e_1[\theta], e_2[\theta] \rangle : T$ is derivable. Note that $e[\theta] = \langle e_1[\theta], e_2[\theta] \rangle$, and we are done. ■

Lemma 3.0.11 *Assume that $\Gamma \vdash e : T$ is derivable. If e is a redex and e' is a reduct of e , then $\Gamma \vdash e' : T$ is also derivable.*

Proof As an exercise. ■

A typing judgment for assigning types to evaluation contexts in \mathcal{L}_0 is of the form $\Gamma \vdash E : T_0/T$, meaning that E can be assigned the type T under Γ if the hole \square in E is given the type T_0 . The rules for deriving such a judgment are given in Figure ??.

Lemma 3.0.12 *If both $\Gamma \vdash E : T_0 \rightarrow T$ and $\Gamma \vdash e : T_0$ are both derivable, then $\Gamma \vdash E[e] : T$ is also derivable.*

Proof As an exercise. ■

Lemma 3.0.13 *If $\Gamma \vdash E[e] : T$ is derivable, then there exists a type T_0 such that both $\Gamma \vdash E : T_0 \rightarrow T$ and $\Gamma \vdash e : T_0$ are both derivable.*

Proof As an exercise. ■

$$\begin{array}{c}
\frac{\begin{array}{c} \vdash c : (T_1, \dots, ty_n) \rightarrow T \quad \Gamma \vdash E : T_0/T \\ \Gamma \vdash v_k : T_k \text{ for } 1 \leq k < i \quad \Gamma \vdash e_k : T_k \text{ for } i < k \leq n \end{array}}{\Gamma \vdash c(v_1, \dots, v_{i-1}, E, e_{i+1}, \dots, e_n) : T_0/T} \text{ (tc-cst)} \\
\frac{\Gamma \vdash E : T_0/bool \quad \Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T}{\Gamma \vdash \mathbf{if}(E, e_1, e_2) : T_0/T} \text{ (tc-if)} \\
\frac{\Gamma \vdash E : T_0/T_1 \quad \Gamma \vdash e : T_2}{\Gamma \vdash \langle E, e \rangle : T_0/T_1 * T_2} \text{ (tc-tup-1)} \\
\frac{\Gamma \vdash e : T_1 \quad \Gamma \vdash E : T_0/T_2}{\Gamma \vdash \langle e, E \rangle : T_0/T_1 * T_2} \text{ (tc-tup-2)} \\
\frac{\Gamma \vdash E : T_0/T_1 * T_2}{\Gamma \vdash \mathbf{fst}(E) : T_0/T_1} \text{ (tc-fst)} \\
\frac{\Gamma \vdash E : T_0/T_1 * T_2}{\Gamma \vdash \mathbf{snd}(E) : T_0/T_2} \text{ (tc-snd)} \\
\frac{\Gamma \vdash e : T_1 \rightarrow T_2 \quad \Gamma \vdash E : T_0/T_1}{\Gamma \vdash (e)E : T_0/T_2} \text{ (tc-app)} \\
\frac{\Gamma \vdash E : T_0/T_1 \rightarrow T_2 \quad \Gamma \vdash e : T_1}{\Gamma \vdash (E)e : T_0/T_2} \text{ (tc-app)}
\end{array}$$

Figure 3.3: The typing rules for evaluation contexts in \mathcal{L}_0

Theorem 3.0.14 (Subject Reduction) Assume that $\Gamma \vdash e_1 : T$ is derivable and $e_1 \rightarrow e_2$ holds. Then $\Gamma \vdash e_2 : T$ is also derivable.

Proof Assume that $e_1 = E[e]$ for some evaluation context E and redex e , and $e_2 = E[e']$ for some reduct e' of e . By Lemma 3.0.13, there exists a type T_0 such that $\Gamma \vdash E : T_0 \rightarrow T$ and $\Gamma \vdash e : T_0$ are derivable. By Lemma 3.0.11, $\Gamma \vdash e' : T_0$ is derivable, and by Lemma 3.0.12, $\Gamma \vdash e_2 : T$ is derivable since $e_2 = E[e']$. ■

Theorem 3.0.15 (Progress) Assume that $\emptyset \vdash e : T$ is derivable. Then either e is value or $e \rightarrow e'$ for some e' .

Proof We proceed by structural induction on the typing derivation \mathcal{D} of $\emptyset \vdash e : T$.

- The last rule applied in \mathcal{D} is **(ty-cst)**. Then \mathcal{D} is of the following form:

$$\frac{\vdash c : (T_1, \dots, T_n) \rightarrow T \quad \mathcal{D}_i :: \Gamma \vdash e_i : T_i \text{ for } 1 \leq i \leq n}{\Gamma \vdash c(e_1, \dots, e_n) : T} \text{ (ty-cst)}$$

where $e = c(e_1, \dots, e_n)$. We have two subcases.

- There exists e_i for some $1 \leq i \leq n$ that is not a value but e_j are values for all $1 \leq j < i$. By induction hypothesis on \mathcal{D}_i , $e_i \rightarrow e'_i$. So we have $e \rightarrow c(e_1, \dots, e_{i-1}, e'_i, e_{i+1}, \dots, e_n)$.
- All e_i are values for $1 \leq i \leq n$. If c is a constant constructor, then e is a value. If c is a constant function, then $e \rightarrow v$ for some v of type T that is a defined value of e .

- The last rule applied in \mathcal{D} is **(ty-if)**. Then \mathcal{D} is of the following form:

$$\frac{\mathcal{D}_0 :: \Gamma \vdash e_0 : \text{bool} \quad \mathcal{D}_1 :: \Gamma \vdash e_1 : T_1 \quad \mathcal{D}_2 :: \Gamma \vdash e_2 : T_2}{\Gamma \vdash \text{if}(e_0, e_1, e_2) : T} \text{ (ty-if)}$$

where $e = \text{if}(e_0, e_1, e_2)$. We have two subcases.

- e_0 is not a value. Then by induction hypothesis on \mathcal{D}_0 , $e_0 \rightarrow e'_0$ holds for some e'_0 . So we have $e \rightarrow \text{if}(e'_0, e_1, e_2)$.
- e_0 is a value. By Lemma 3.0.9, e_0 is either *true* or *false*. If e_0 is *true*, then $e \rightarrow e_1$ holds. Otherwise, e_0 is *false*, and $e \rightarrow e_2$ holds.

- The last rule applied in \mathcal{D} is **(ty-fst)**. Then \mathcal{D} is of the following form:

$$\frac{\mathcal{D}_0 :: \Gamma \vdash e_0 : T_1 * T_2}{\Gamma \vdash \text{fst}(e_0) : T_1} \text{ (ty-fst)}$$

where $e = \text{fst}(e_0)$ and $T = T_1$. We have two subcases.

- e_0 is not a value. By induction hypothesis on \mathcal{D}_0 , $e_0 \rightarrow e'_0$ for some e'_0 . So we have $e \rightarrow \text{fst}(e'_0)$.
- e_0 is a value. By Lemma 3.0.9, e_0 is of the form $\langle v_1, v_2 \rangle$. So we have $e \rightarrow v_1$.

- The last rule applied in \mathcal{D} is **(ty-fst)**. Then this case is symmetric to the previous one.
- The last rule applied in \mathcal{D} is **(ty-lam)**. Then e is a value.
- The last rule applied in \mathcal{D} is **(ty-app)**. Then \mathcal{D} is of the following form:

$$\frac{\mathcal{D}_1 :: \Gamma \vdash e_1 : T_1 \rightarrow T_2 \quad \mathcal{D}_2 :: \Gamma \vdash e_2 : T_1}{\Gamma \vdash (e_1)e_2 : T_2} \text{ (ty-app)}$$

where $e = (e_1)e_2$ and $T = T_2$. We have a few subcases.

- e_1 is not a value. By induction hypothesis on \mathcal{D}_1 , $e_1 \rightarrow e'_1$. Therefore, $e \rightarrow (e'_1)e_2$.
 - e_1 is a value but e_2 is not a value. By induction hypothesis on \mathcal{D}_2 , $e_2 \rightarrow e'_2$. Therefore, $e \rightarrow (e_1)e'_2$.
 - e_1 and e_2 are values. By Lemma 3.0.9, e_1 is of the form $\lambda x.e_{10}$, and thus we have $e \rightarrow e_{10}[x := e_2]$.
- The last rule applied in \mathcal{D} is **(ty-fix)**. Then \mathcal{D} is of the following form:

$$\frac{\Gamma, f : T \vdash e_0 : T}{\Gamma \vdash \mathbf{fix} f.e_0 : T} \text{ (ty-fix)}$$

where $e = \mathbf{fix} f.e_0$. Then $e \rightarrow e_0[f := \mathbf{fix} f.e_0]$.

We conclude the proof as all the cases are covered. ■

3.1 Normalization

Definition 3.1.1 (Reducibility Predicates) Given a simple type T , a predicate \hat{R}_T on values in \mathcal{L}_0 and another predicate R_T on expressions in \mathcal{L}_0 are defined below by structural induction on T .

- $R_T(e)$ holds if $e \downarrow$ and for every value v , $e \rightarrow^* v$ implies $\hat{R}_T(v)$.
- T is some base type δ . Then $\hat{R}_T(v)$ holds for every value v .
- $T = T_1 * T_2$ for some types T_1 and T_2 . Then $\hat{R}_T(\langle v_1, v_2 \rangle)$ holds if both $\hat{R}_{T_1}(v_1)$ and $\hat{R}_{T_2}(v_2)$.
- $T = T_1 \rightarrow T_2$ for some types T_1 and T_2 . Then $\hat{R}_T(\lambda x.e_0)$ holds if for every value v , $\hat{R}_{T_1}(v)$ implies $R_{T_2}(e_0[x := v])$.

Lemma 3.1.2 We have the following.

1. Given T, e and e' , if $R_T(e)$ and $e \rightarrow e'$, then $R_T(e')$.
2. Given T and e , where e is not a value, if $e \rightarrow e'$ implies $R_T(e')$ for every e' , then $R_T(e)$.

Proof The lemma follows from the definition of reducibility predicates immediately. ■

Lemma 3.1.3 We have the following.

1. If $R_{T_1}(e_1)$ and $R_{T_2}(e_2)$, then $R_{T_1 * T_2}(\langle e_1, e_2 \rangle)$.
2. If $\hat{R}_{T_1}(v)$ implies $R_{T_2}(e_0[x := v])$ for every value v , then $R_{T_1 \rightarrow T_2}(\lambda x. e_0)$.

Proof We prove (1) by induction on $\mu(e_1) + \mu(e_2)$. If e is a value, then $\hat{R}_{T_1 * T_2}(e)$ holds by the definition of reducibility predicates, which implies $R_{T_1 * T_2}(e)$. If e is not a value and $e \rightarrow e'$ holds, then there are two possibilities:

- $e' = \langle e'_1, e'_2 \rangle$ for some e'_1 such that $e_1 \rightarrow e'_1$. Clearly, we have $\mu(e_1) < \mu(e'_1)$. By Lemma 3.1.2 (1), $R_T(e'_1)$ holds. So we have $R_{T_1 * T_2}(e')$ by induction hypothesis.
- $e' = \langle e_1, e'_2 \rangle$ for some e'_2 such that $e_2 \rightarrow e'_2$. This case is handled in the same manner as the previous one.

By Lemma 3.1.2 (2), $R_{T_1 * T_2}(e)$ holds. Hence, the proof for (1) is concluded. Clearly, (2) follows from the definition of reducibility predicated immediately. ■

Lemma 3.1.4 Assume that $R_{T_0}(e_0)$ holds.

1. If $T_0 = \text{bool}$, then for every T, e_1 and e_2 , $R_T(e_1)$ and $R_T(e_2)$ implies $R_T(\text{if}(e_0, e_1, e_2))$.
2. If $T_0 = T_1 * T_2$, then $R_{T_1}(\text{fst}(e_0))$ and $R_{T_2}(\text{snd}(e_0))$.
3. If $T_0 = T_1 \rightarrow T_2$, then $R_{T_1}(e_1)$ implies $R_{T_2}((e_0)e_1)$ for every e_1 .

Proof We prove (1) by induction on $\mu(e_0)$. Assume that $e = \text{if}(e_0, e_1, e_2) \rightarrow e'$. Then there are three possibilities:

- $e' = \text{if}(e'_0, e_1, e_2)$ for some e'_0 such that $e_0 \rightarrow e'_0$ holds. Clearly, we have $\mu(e'_0) < \mu(e_0)$. By Lemma 3.1.2 (1), $R_T(e'_0)$ holds. Thus, we have $R_T(e')$ by induction hypothesis.
- $e_0 = \text{true}$ and $e' = e_1$. By assumption, $R_T(e')$ holds.
- $e_0 = \text{false}$ and $e' = e_2$. By assumption, $R_T(e')$ holds.

Thus, we have $R_T(e)$ by Lemma 3.1.2 (2). Both (2) and (3) can be proven similarly. ■

Lemma 3.1.5 Assume that $\Gamma \vdash e : T$ is derivable and θ is a substitution such that $\text{dom}(\theta) = \text{dom}(\Gamma)$ and $R_{\Gamma(x)}(\theta(x))$ for every $x \in \text{dom}(\theta)$. Then $R_T(e[\theta])$ holds.

Proof The proof proceeds by structural induction on the typing derivation \mathcal{D} of $\Gamma \vdash e : T$.

- The last applied rule in \mathcal{D} is (**ty-var**):

$$\frac{\Gamma(x) = T}{\Gamma \vdash x : T}$$

where $e = x$. Then $e[\theta] = \theta(x)$, and we have $R_{\Gamma(x)}(e[\theta])$.

- The last applied rule in \mathcal{D} is **(ty-lam)**:

$$\frac{\mathcal{D}_1 :: \Gamma, x : T_1 \vdash e_1 : T_2}{\Gamma \vdash \lambda x. e_1 : T_1 \rightarrow T_2}$$

where $e = \lambda x. e_1$ and $T = T_1 \rightarrow T_2$. Clearly, $e[\theta] = \lambda x. e_1[\theta]$. Assume that v is a value satisfying $R_{T_1}(v)$ and $\theta_1 = \theta[x \mapsto v]$. By induction hypothesis on \mathcal{D}_1 , we have $R_{T_2}(e_1[\theta_1])$. Note that $e_1[\theta_1] = e_1[\theta][x := v]$. By Lemma 3.1.3 (2), we have $R_T(\lambda x. e_1[\theta])$. Hence, $R_T(e[\theta])$ holds.

The rest of the cases can be handled similarly. ■

Theorem 3.1.6 *Assume that $\emptyset \vdash e : T$ is derivable. Then $e \downarrow$ holds.*

Proof By Lemma 3.1.5, we have $R_T(e[\theta])$ for the empty substitution θ . Therefore, $e \downarrow$ holds. ■