Typing Local Control and State using Flow Analysis

Elided Proofs

Arjun Guha, Claudiu Saftoiu, and Shriram Krishnamurthi

October 8, 2010

Chapter 1

Proofs and Auxilliary Definitions

1.1 Full Typing Relation

$$\begin{aligned} ty_1(\texttt{typeof}) &= \top \to \mathsf{Str} \\ ty_2(\texttt{===}) &= \top \times \top \to \mathsf{Bool} \\ ty_2(\texttt{-}) &= \mathsf{Num} \times \mathsf{Num} \to \mathsf{Bool} \end{aligned}$$

$$\frac{\Sigma; \Gamma \vdash e : S \cdots \qquad ty_n(op_n) = S \cdots \to T}{\Sigma; \Gamma \vdash op_n\texttt{(}e \cdots\texttt{)} : T} \quad \text{(T-PrimApp)}$$

$$\Sigma; \Gamma \vdash num : \mathsf{Num} \quad \text{(T-Num)}$$

$$\Sigma; \Gamma \vdash str : \mathsf{Str} \quad \text{(T-Str)}$$

$$\Sigma; \Gamma \vdash bool : \mathsf{Bool} \quad \text{(T-Bool)}$$

$$\Sigma; \Gamma \vdash \texttt{undefined} : \mathsf{Undef} \quad \text{(T-Undef)}$$

$$\frac{\Sigma(l) = T}{\Sigma; \Gamma \vdash l : T} \quad \text{(T-Loc)}$$

$$\frac{\Gamma(x) = T}{\Sigma; \Gamma \vdash x : T} \quad \text{(T-Id)}$$

$$\frac{\Sigma; \Gamma', x : S, \cdots \vdash e : T \qquad \Gamma' = \Gamma \text{ with labels removed}}{\Sigma; \Gamma \vdash \texttt{func(}x \cdots\texttt{) :}S \cdots \to T\texttt{\{ }e\texttt{ \}} : S \cdots \to T} \quad \text{(T-Abs)}$$

$$\frac{\Sigma; \Gamma \vdash e : S \cdots \qquad \Sigma; \Gamma \vdash f : S \cdots \to T}{\Sigma; \Gamma \vdash f\texttt{(}e \cdots\texttt{)} : T} \quad \text{(T-App)}$$

$$\frac{\Sigma;\Gamma \vdash e_1 : S \qquad \Sigma; x : S, \Gamma \vdash e_2 : T}{\Sigma;\Gamma \vdash \texttt{let}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 : T} \tag{T-Let}$$

$$\frac{\Sigma;\Gamma \vdash e : T}{\Sigma;\Gamma \vdash \texttt{ref}\ e : \mathsf{Ref}\ T} \tag{T-Ref}$$

$$\frac{\Sigma;\Gamma \vdash e : \mathsf{Ref}\ T}{\Sigma;\Gamma \vdash \texttt{deref}\ e : T} \tag{T-Deref}$$

$$\frac{\Sigma;\Gamma \vdash e_1 : \mathsf{Ref}\ S \qquad \Sigma;\Gamma \vdash e_2 : T \qquad T <: S}{\Sigma;\Gamma \vdash \texttt{setref}\ e_1\ e_2 : \mathsf{Ref}\ \ T} \tag{T-SetRef}$$

$$\frac{\Sigma;\Gamma \vdash e_1 : \mathsf{Bool} \qquad \Sigma;\Gamma \vdash e_2 : T \qquad \Sigma;\Gamma \vdash e_3 : T}{\Sigma;\Gamma \vdash \texttt{if (}e_1\texttt{) \{ }e_2\texttt{ \} else \{ }e_3\texttt{ \}} : T} \tag{T-If}$$

$$\frac{\Sigma;\Gamma, \mathit{label} : T \vdash e : T}{\Sigma;\Gamma \vdash \mathit{label}\texttt{:}T\ \texttt{\{ }e\texttt{ \}} : T} \tag{T-Label}$$

$$\frac{\Gamma(\mathit{label}) = T \qquad \Sigma, \Gamma \vdash e : T}{\Sigma;\Gamma \vdash \texttt{break}\ \mathit{label}\ e : \bot} \tag{T-Break}$$

$$\frac{\Sigma;\Gamma \vdash e : S \qquad S <: T}{\Sigma;\Gamma \vdash e : T} \tag{T-Sub}$$

$$\frac{\forall l \in \mathit{dom}(\Sigma).\Sigma;\cdot \vdash \sigma(l) : \Sigma(l) \qquad \mathit{dom}(\Sigma) = \mathit{dom}(\sigma)}{\Sigma \vdash \sigma} \tag{T-σ}$$

1.2 Type Safety

Lemma 1 (Progress) *If $\Sigma, \cdot \vdash e : T$ and $\Sigma \vdash \sigma$ then either:*

i. $e \in v$, or

ii. There exists σ', e', such that $\sigma e \to \sigma' e'$, or

iii. There exists an E, such that $e = E\langle \textsf{tagerr} \rangle$.

Proof: By lemma 3, cases (i.) and (iii.) are immediate, leaving only the the case where $e = E\langle ae \rangle$. By lemma 4, there exist S and Γ, such that $\Sigma;\Gamma \vdash ae : S$. By case-analysis on ae, applying inversion (lemma 8) and canonical forms (lemma 9) where indicated:

- Case $ae = \textsf{ref}\ v$. By E-Ref, $\sigma E\langle \textsf{ref}\ e \rangle \to \sigma, (l, v)\langle l \rangle$ where $l \notin \mathit{dom}(\sigma)$.

- Case $ae = \textsf{deref}\ v$. By inversion, $\Sigma; \Gamma \vdash v : \textsf{Ref}\ \ S'$. By canonical forms, $v = l$ and $l \in dom(\Sigma)$. By T-σ, $l \in dom(\sigma)$. By E-Deref, $\sigma E\langle \textsf{deref}\ \mathsf{l} \rangle \rightarrow \sigma E\langle \sigma(l) \rangle$.

- Case $ae = \textsf{setref}\ v_1\ v_2$. By inversion, $\Sigma; \Gamma \vdash v_1 : \textsf{Ref}\ \ S_1$. By canonical forms, $v_1 = l$, hence $l \in dom(\sigma)$. By E-SetRef, $\sigma E\langle \textsf{setref}\ l\ v_2 \rangle \rightarrow \sigma[l := v_2] E\langle l \rangle$.

- Case $ae = label\texttt{:}T\texttt{\{}\ E_1\langle \textsf{break}\ label\ v \rangle\ \texttt{\}}$. By E-Break, $\sigma E\langle label\texttt{:}T\texttt{\{}\ E_1\langle \textsf{break}\ label\ v \rangle\ \texttt{\}} \rangle \rightarrow \sigma E\langle v \rangle$.

- Case $ae = label\texttt{:}T\ \texttt{\{}\ v\ \texttt{\}}$. By E-Label-Pop, $\sigma E\langle label\texttt{:}T\ \texttt{\{}\ v\ \texttt{\}} \rangle \rightarrow \sigma E\langle v \rangle$.

- Case $ae = \textsf{let}\ x\ \texttt{=}\ v\ \textsf{in}\ e$. By E-Let, $\sigma E\langle \textsf{let}\ x\ \texttt{=}\ v\ \textsf{in}\ e \rangle \rightarrow \sigma E\langle e[x/v] \rangle$.

- Case $ae = v_f\texttt{(}v \cdots\texttt{)}$. By canonical forms, $\Sigma; \Gamma \vdash v_f : S' \cdots \rightarrow S$. By inversion (lemma 8), $v_f = \textsf{func}\texttt{(}x \cdots\texttt{)}\ \texttt{\{}\ e\ \texttt{\}}$. By E-Abs, $\sigma E\langle \textsf{func}\texttt{(}x \cdots\texttt{)}\ \texttt{\{}\ e\ \texttt{\}}\texttt{(}v \cdots\texttt{)} \rangle \rightarrow \sigma E\langle e'[x/v] \cdots \rangle$.

- Case $ae = op_n\texttt{(}v \cdots\texttt{)}$ is similar to the case above.

- Case $ae = \textsf{tagcheck}\ R\ v$. If $\delta_1(\textsf{typeof}, v) \in R$, then $\sigma E\langle \textsf{tagcheck}\ R\ v \rangle \rightarrow \sigma E\langle v \rangle$ by E-TagCheck. If not, then $\sigma E\langle \textsf{tagcheck}\ R\ v \rightarrow \sigma E\langle \textsf{tagerr} \rangle$ by E-TagCheck-Err.

■

Lemma 2 (Preservation) *If $\Sigma_1; \cdot \vdash e_1 : T$, $\Sigma_1 \vdash \sigma_1$, and $\sigma_1 e_1 \rightarrow \sigma_2 e_2$, then there exists a Σ_2, such that:*

i. $\Sigma_2; \cdot \vdash e_2 : T$,

ii. $\Sigma_2 \vdash \sigma_2$, *and*

iii. $\Sigma_1 \subseteq \Sigma_2$.

Proof: By case-analysis of the reduction rules, there exists an evaluation context (E), an active expression (ae), and an expression (e'), such that $e_1 = E\langle ae \rangle$ and $e_2 = E\langle e' \rangle$. Hence,

$$\sigma_1 E\langle ae \rangle \rightarrow \sigma_2 E\langle e' \rangle$$

By lemma 4, there exist Γ and S, such that $\Sigma_1; \Gamma \vdash ae : S$ and Γ only contains labels. We will prove that $\Sigma_2; \Gamma \vdash e' : S$, so that $\Sigma_2; \Gamma \vdash E\langle e' \rangle : T$.

We proceed by case-analysis on ae, using inversion (lemma 8) where specified:

- $\textsf{let}\ x\ \texttt{=}\ v\ \textsf{in}\ e$. By inversion, $\Sigma_1; \Gamma \vdash v : U$ and $\Sigma_1; x : U, \Gamma \vdash e : S$. By E-Let, $ae \rightarrow e[x/v]$. By substitution (lemma 7), $\Sigma; \Gamma \vdash e[x/v] : S$.

- $\texttt{(}\textsf{func}\ \texttt{(}x \cdots\texttt{)}\ \texttt{\{}\ e\ \texttt{\})(}v \cdots\texttt{)}$. By inversion and canonical forms, $\Sigma_1; \Gamma \vdash v : U \cdots$ and $\Sigma_1; x : U \cdots \vdash e : S$. By E-App, $\sigma\texttt{(}\textsf{func}\ \texttt{(}x \cdots\texttt{)}\ \texttt{\{}\ e\ \texttt{\})(}v \cdots\texttt{)} \rightarrow \sigma e[x/v] \cdots$. By substitution (lemma 7), $\Sigma_1; \cdot \vdash e[x/v] \cdots : S$.

- $label{:}S'$ `{` v `}`. By inversion, $\Sigma_1; label : S', \Gamma \vdash v : S'$ and $S' <: S$. By E-Label-Pop, $\sigma label{:}S\texttt{\{}\ v\ \texttt{\}} \rightarrow \sigma v$. Typing values does not require labels in the environment (lemma 6), hence $\Sigma_1; \Gamma \vdash v : S'$. The conclusion follows by T-Sub.

- $label{:}S'$ `{` $E'\langle$**break** $label\ v\rangle$`}`. By inversion, $\Sigma_1; label : S', \Gamma \vdash E'\langle$**break** $label\ v\rangle : S'$ and $S' <: S$. By E-Break, $\sigma E'\langle$**break** $label\ v\rangle \rightarrow \sigma v$. Since the evaluation context, E' cannot bind identifiers and values can be typed without labels (lemma 6), $\Sigma_1; \Gamma \vdash v : S'$.

- **ref** v. By inversion, $\Sigma_1; \Gamma \vdash v : U$ and Ref $U <: S$. By E-Ref, for an $l \notin dom(\sigma_1)$, $\sigma_2 = \sigma_1, (l, v)$ and σ_1**ref** $v \rightarrow \sigma_2 l$. Let $\Sigma_2 = l : U, \Sigma_1$. By T-Loc, $\Sigma_2; \cdot \vdash l :$ Ref U, followed by T-Sub with Ref $U <: S$.

- **deref** l. By inversion, $\Sigma_1; \Gamma \vdash l :$ Ref U with $U <: S$. By hypothesis and T-Loc, $\Sigma_1(l) = U$. By $\Sigma_1 \vdash \sigma_1$, there exists a v such that $\sigma_1(l) = v$ and $\Sigma; \Gamma \vdash v : U$. By E-Deref, σ_1**deref** $l \rightarrow \sigma_1 v$. By T-Sub, $\Sigma_1; \Gamma \vdash v : S$.

- **setref** $l\ v$. By inversion, $\Sigma_1; \Gamma \vdash l :$ Ref S', $\Sigma_1; \Gamma \vdash v : U$, $U <: S'$, and Ref $S' <: S$. By T-Loc, $\Sigma_1(l) = S'$. By E-SetRef, σ_1**setref** $l\ v \rightarrow \sigma_1[l/v]l$. Since Γ does not bind identifiers and by lemma 6, $\Sigma_1; \cdot \vdash v : U$. By T-Sub, $\Sigma; \cdot \vdash v : S'$. Hence, $\Sigma_1 \vdash \sigma_1[l/v]$.

- **tagcheck** $R\ v$.

  - $\sigma_1 E\langle$**tagcheck** $R\ v\rangle \rightarrow \sigma_1 E\langle v\rangle$, and $\delta_1($**typeof**$, v) \in R$. By inversion, $\Sigma_1; \Gamma \vdash v : U$ and $S' = static(R, U)$ for $S' <: S$. By definition of *static*, $S' <: U$. By case-analysis of v:

    * $v =$ **func** `(`$x \cdots$`)` `:`$T_a \cdots \rightarrow T_r$`{` e `}`, hence $\Sigma_1; \cdot \vdash v : T_a \cdots \rightarrow T_r$ and $T_a \cdots \rightarrow T_r <: U$. By lemma 10, $static(R, T_a \cdots \rightarrow T_r) <: static(R, U) <: S$. By lemma 11, `"function"` $\in runtime(U)$. Hence, $static(R, , T_a \cdots \rightarrow T_r) = T_a \cdots \rightarrow T_r$.
    * Constants are symmetric to functions.

  - $\sigma_1 E\langle$**tagcheck** $R\ v\rangle \rightarrow \sigma_1 E\langle$**tagerr**$\rangle$. Apply T-TagErr and T-Sub.

■

Definition 1 (Active Expressions) *An active expression, ae is one of:*

$$
\begin{array}{rcl}
ae & = & \textbf{let}\ x\ \texttt{=}\ v\ \textbf{in}\ e \\
& | & v_f\ \texttt{(}v \cdots\texttt{)} \\
& | & op_n\ \texttt{(}v \cdots\texttt{)} \\
& | & label{:}T\ \texttt{\{}\ v\ \texttt{\}} \\
& | & \textbf{ref}\ v \\
& | & \textbf{deref}\ v \\
& | & \textbf{setref}\ v_1\ v_2 \\
& | & label{:}T\texttt{\{}\ E\langle\textbf{break}\ v\rangle\ \texttt{\}}\ \textit{where}\ label \notin E \\
& | & \textbf{tagcheck}\ R\ v
\end{array}
$$

Lemma 3 *For all closed expressions e, either:*

i. $e \in v$,

ii. there exist e *and* ae *such that* $e = E\langle ae \rangle$, *or*

iii. $e = E\langle \textsf{tagerr} \rangle$, *for some* E.

Proof: By definition of e, v, and E. ■

Lemma 4 (Closed Active Expressions) *For all* Σ, T, E, ae, *if* $\Sigma; \cdot \vdash E\langle ae \rangle : T$, *then there exist* S *and* Γ, *where* Γ *only contains labels, such that* $\Sigma; \Gamma \vdash ae : S$.

Proof: there exists a subdeduction of the typing derivation, such that $\Sigma; \Gamma \vdash ae : S$. Suppose an identifier, $x \in dom(\Gamma)$. Then, ae is in a program context $C\langle \textsf{let } x \texttt{ = } e_1 \textsf{ in } \bullet \rangle$ or $C\langle \textsf{func}(x \cdots) \texttt{ \{ } \bullet \texttt{ \}} \rangle$. However, this cannot occur since ae is in an evaluation context. ■

Lemma 5 *For all* E, *if* $\sigma_1 e_1 \to \sigma_2 e_2$, *then* $\sigma_1 E\langle e_1 \rangle \to \sigma_2 E\langle e_2 \rangle$.

Proof: by case-analysis of the reduction rules, there exist E', e_1', e_2', such that $e_1 = E'\langle e_1' \rangle$, $e_2 = E'\langle e_2' \rangle$, and $\sigma_1 E'\langle e_1' \rangle \to \sigma_2 E'\langle e_2' \rangle$. By the same reduction rule, $\sigma_1 E\langle E'\langle e_1' \rangle \rangle \to \sigma_2 E\langle E'\langle e_2' \rangle \rangle$. ■

Lemma 6 *If* $\Sigma; \Gamma \vdash v : T$ *then* $\Sigma; \Gamma' \vdash v : T$, *where* $\Gamma' = \Gamma$ *with labels removed.*

Proof: the only interesting case is $v = \textsf{func}(x \cdots) \texttt{ \{ } e \texttt{ \}}$, where e is typed in an extension of Γ with labels removed, i.e., Γ'. ■

Lemma 7 (Substitution) *If* $\Sigma; x : S, \Gamma \vdash e : T$ *and* $\Sigma; \Gamma \vdash v : S$, *then* $\Sigma; \Gamma \vdash e[x/v] : T$.

Proof: by induction on the typing derivation. ■

Lemma 8 (Inversion) *If:*

- $\Sigma; \Gamma \vdash \textsf{ref } e : T$, *then* $\textsf{Ref } S <: T$ *and* $\Sigma; \Gamma \vdash e : S$,
- $\Sigma; \Gamma \vdash \textsf{deref } e : T$, *then* $\Sigma; \Gamma \vdash e : \textsf{Ref } S$ *with* $S <: T$,
- $\Sigma; \Gamma \vdash \textsf{setref } e_1 \ e_2 : T$, *then* $\Sigma; \Gamma \vdash e_1 : \textsf{Ref } S$, $\Sigma; \Gamma \vdash e_2 : U$, $U <: S$, *and* $\textsf{Ref } S <: T$,
- $\Sigma; \Gamma \vdash e_f(e \cdots) : T$, *then* $\Sigma; \Gamma \vdash e_f : S \cdots \to T'$, $\Sigma; \Gamma \vdash e : S \cdots$, *and* $T' <: T$.
- $\Sigma; \Gamma \vdash label : S \texttt{ \{ } e \texttt{ \}} : T$, *then* $S <: T$ *and* $\sigma; label : S, \Gamma \vdash e : S$.
- $\Sigma; \Gamma \vdash \textsf{let } x \texttt{ = } e_1 \textsf{ in } e_2 : T$, *then* $\Sigma; \Gamma \vdash e_1 : S$ *and* $\Sigma; x : S, \Gamma \vdash e_2 : T$.
- $\Sigma; \Gamma \vdash \textsf{tagcheck } R \ e : T$, *then* $\Sigma; \Gamma \vdash e = S$ *and* $static(R, S) = T'$, *where* $T' <: T$.

Proof: by induction on the typing derivation. In all cases, only T-Sub and one other typing rule apply. T-Sub requires induction and the other latter is immediate.

- Only T-Ref and T-Sub apply. T-Ref is immediate with $\mathsf{Ref}\ \ S = T$. For T-Sub, by inversion, $\Sigma; \Gamma \vdash \textsf{ref e} : T'$ and $T' <: T$. By induction, $T' = \mathsf{Ref}\ \ S$ and $\Sigma; \Gamma \vdash e : S$.
- Only T-Deref and T-Sub apply. T-Deref is immediate with $T <: T$. For case T-Sub, by inversion $\Sigma; \Gamma \vdash \textsf{deref e} : S$ and $S <: T$. By induction, $\Sigma; \Gamma \vdash e : \mathsf{Ref}\ \ S'$ with $S' <: S$. By S-Trans, $S' <: T$
- Only T-SetRef and T-Sub apply. T-SetRef is immediate, with $\mathsf{Ref}\ \ S = T$. For T-Sub, by inversion, $\Sigma; \Gamma \vdash \textsf{setref}\ e_1\ e_2 : T'$ and $T' <: T$. By induction, $\Sigma; \Gamma \vdash e_1 : \mathsf{Ref}\ \ S$, $\Sigma; \Gamma \vdash e_2 : U$, $U <: S$, and $\mathsf{Ref}\ \ S <: T'$. By S-Trans, $\mathsf{Ref}\ \ S <: T$.
- Only cases T-App and T-Sub apply. T-App is immediate. For T-Sub, the conclusion follows by induction and S-Arr.
- Only cases T-Sub and T-Label apply. T-Label follows immediately with $S = T$ and S-Refl. For T-Sub, $\Sigma; \Gamma \vdash \mathit{label}\texttt{:}S\ \texttt{\{}\ e\ \texttt{\}} : T'$ and $T' <: T$. By induction, $\Sigma; \mathit{label} : S, \Gamma \vdash e : S$ and $S <: T'$. Hence, by S-Trans, $S <: T$.
- Only T-Let and T-Sub apply. T-Let is immediate. For T-Sub, by induction, $\Sigma; \Gamma \vdash e_1 : S$, $\Sigma; x : S, \Gamma \vdash e_2 : T'$, and $T' <: T$. Apply T-Sub.
- Only T-TagCheck and T-Sub apply. T-TagCheck is immediate. For T-Sub, by induction, it holds for a $T' <: T$.

■

Lemma 9 (Canonical Forms) *For $\Sigma; \Gamma \vdash v : T$:*

i. If $T = \mathsf{Ref}\ \ S$ then $v = loc$ and $\Sigma(loc) <: S$.

ii. If $T = U \cdots \to S$ then $v = \textsf{func}\texttt{(}x \cdots\texttt{)}\ \texttt{\{}\ e\ \texttt{\}}$.

Proof: By induction on the typing derivation.

i. Only T-Loc and T-Sub apply. T-Loc is immediate. For T-Sub, $\Sigma; \Gamma \vdash v : T'$ with $T' <: T$. By induction, $v = loc$, $T' = \mathsf{Ref}\ \ S'$, and $\Sigma(loc) <: S'$.

ii. Only T-Abs and T-Sub apply. T-Abs is immediate, and T-Sub follows by induction.

■

Lemma 10 (*static* commutes with subtyping) *If $S <: T$, then $\mathit{static}(R, S) <: \mathit{static}(R, T)$.*

Proof: by induction on $S <: T$. ■

Lemma 11 *If $\Sigma; \Gamma \vdash v : T$, then $\delta_1(\textsf{typeof}, v) \in \mathit{runtime}(T)$.*

Proof: by case analysis of v. ■

1.3 CPS Transformation

The following four mutually-recursive functions define our CPS transformation: $\mathcal{C}_k$ transforms expressions, Φ transforms values, $\mathcal{K}_k$ transforms evaluation contexts, and $\mathcal{P}_k$ transforms program states (expressions with stores). This style of transformation is due to Sabry and Felleisen [1]. Our presentation has a few minor distinctions:

- The semantics of our source language directly specifies the evaluation of **break**, which is a very restricted form of *call/cc*. Hence, **break** is an expression transformed by $\mathcal{C}_k$, instead of a value transformed by Φ.
- We use a global store instead of a syntactic store, hence the need for $\mathcal{P}_k$.
- We use **let**-expressions to avoid continuation-passing operators.
- We do not employ a fully-compacting transformation. For example, we do not have a special-case for $\mathcal{C}_k[\![E\langle\texttt{(}\textsf{func}\texttt{(}x\cdots\texttt{):T \{ }e\texttt{ \})(}v\cdots\texttt{)}\rangle]\!]$, as it would introduce interprocedural flows in the source language. (In contrast, see [1, Definition 5].)

$\boxed{\mathcal{C}_k : e \to M}$

$$
\begin{array}{rcl}
\mathcal{C}_k[\![v]\!] & = & k(\Phi[\![v]\!]) \\
\mathcal{C}_k[\![E\langle\textsf{let (}x\texttt{ = }v\textsf{) }e\rangle]\!] & = & \textsf{let (}x\texttt{ = }\Phi[\![v]\!]\textsf{)}\mathcal{C}_k[\![E\langle e\rangle]\!] \\
\mathcal{C}_k[\![E\langle v_f\texttt{(}v\cdots\texttt{)}\rangle]\!] & = & \Phi[\![v_f]\!]\texttt{(}\mathcal{K}_k[\![E]\!]\texttt{,}\Phi[\![v]\!]\cdots\texttt{)} \\
\mathcal{C}_k[\![E\langle op_n\texttt{(}v\cdots\texttt{)}\rangle]\!] & = & \textsf{let }x\texttt{ = }op_n\texttt{(}\Phi[\![v]\!]\cdots\texttt{) }\textsf{in }\mathcal{K}_k[\![E]\!]\texttt{(}x\texttt{)} \\
\mathcal{C}_k[\![E\langle\textsf{if (}v\textsf{) \{ }e_2\textsf{ \} else \{ }e_3\textsf{ \}}\rangle]\!] & = & \textsf{(func (}k\textsf{) \{ if (}\Phi[\![v]\!]\textsf{) \{ }\mathcal{C}_k[\![e_2]\!]\textsf{ \} else \{ }\mathcal{C}_k[\![e_3]\!]\textsf{ \} \})(}\mathcal{K}_k[\![E]\!]\textsf{)} \\
\mathcal{C}_k[\![E_1\langle label\texttt{: \{ }E_2\langle\textsf{break }label\ v\rangle\texttt{ \}}\rangle]\!] & = & \mathcal{K}_k[\![E_1]\!]\texttt{(}\Phi[\![v]\!]\texttt{)}, \text{ when } label \notin E_2 \\
\mathcal{C}_k[\![E\langle label\texttt{:\{ }v\texttt{ \}}\rangle]\!] & = & \mathcal{C}_k[\![E\langle v\rangle]\!] \\
\mathcal{C}_k[\![E\langle\textsf{ref }v\rangle]\!] & = & \textsf{let }x\texttt{ = }\textsf{ref }\Phi[\![v]\!]\textsf{ in }\mathcal{K}_k[\![E]\!]\texttt{(}x\texttt{)} \\
\mathcal{C}_k[\![E\langle\textsf{deref }v\rangle]\!] & = & \textsf{let }x\texttt{ = }\textsf{deref }\Phi[\![v]\!]\textsf{ in }\mathcal{K}_k[\![E]\!]\texttt{(}x\texttt{)} \\
\mathcal{C}_k[\![E\langle\textsf{setref }l\ v\rangle]\!] & = & \textsf{let }x\texttt{ = }\textsf{setref }l\ \Phi[\![v]\!]\textsf{ in }\mathcal{K}_k[\![E]\!]\texttt{(}l\texttt{)} \\
\mathcal{C}_k[\![E\langle\textsf{tagcheck }R\ v\rangle]\!] & = & \textsf{let }x\texttt{ = }\textsf{tagcheck }R\ \Phi[\![v]\!]\textsf{ in }\mathcal{K}_k[\![E]\!]\texttt{(}x\texttt{)}
\end{array}
$$

$\boxed{\Phi : v \to V}$

$$
\begin{array}{rcl}
\Phi[\![x]\!] & = & x \\
\Phi[\![\textsf{func(}x\cdots\textsf{):}S\cdots\to T\textsf{ \{ }e\textsf{ \}}]\!] & = & \textsf{func(}k\textsf{,}x\cdots\textsf{):}(T\to\bot)\times S\cdots\to\bot\textsf{ \{ }\mathcal{C}_k[\![e]\!]\textsf{ \}} \\
\Phi[\![c]\!] & = & c \\
\Phi[\![l]\!] & = & l
\end{array}
$$

$\boxed{\mathcal{K}_k : E \to V}$

$$
\begin{array}{rcl}
\mathcal{K}_k[\![\bullet]\!] & = & k \\
\mathcal{K}_k[\![E\langle \textsf{let (}x\textsf{ = }\bullet\textsf{) }e\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle e\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \bullet\textsf{(}e\cdots\textsf{)}\rangle]\!] & = & \textsf{func(}f\textsf{) \{ }\mathcal{C}_k[\![E\langle f\textsf{(}e\cdots\textsf{)}\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle v_f\textsf{(}v\cdots\bullet\, e\cdots\textsf{)}\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle v_f\textsf{(}v\cdots x e\cdots\textsf{)}\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle op_n\textsf{(}v\cdots\bullet\, e\cdots\textsf{)}\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle op_n\textsf{(}v\cdots x e\cdots\textsf{)}\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \textsf{if (}\bullet\textsf{) \{ }e_2\textsf{ \} else \{ }e_3\textsf{ \}}\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle \textsf{if (}x\textsf{) \{ }e_2\textsf{ \} else \{ }e_3\textsf{ \}}\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \textsf{ref }\bullet\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle \textsf{ref }x\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \textsf{deref }\bullet\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle \textsf{deref }x\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \textsf{setref }\bullet\ e\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle x\textsf{ = }e\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \textsf{setref }v\ \bullet\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle v\textsf{ = }x\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \textsf{break }label\ \bullet\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle \textsf{break }label\ x\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle label\textsf{: \{ }\bullet\textsf{ \}}\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle label\textsf{: \{ }x\textsf{ \} }\rangle]\!]\textsf{ \}} \\
\mathcal{K}_k[\![E\langle \textsf{tagcheck }R\ \bullet\rangle]\!] & = & \textsf{func(}x\textsf{) \{ }\mathcal{C}_k[\![E\langle \textsf{tagcheck }R\ x\rangle]\!]\textsf{\}}
\end{array}
$$

$\boxed{\mathcal{P}_k : \sigma e \to SM}$

$$\mathcal{P}_l[\![(l,v)\cdots e]\!] \quad = \quad (l, \Phi(v))\cdots \mathcal{C}_k[\![e]\!]$$

Lemma 12 (Soundness of $\mathcal{P}_k$) *If $\sigma e \to_R \sigma' e'$, then $\mathcal{P}_k[\![\sigma e]\!] \twoheadrightarrow_{\{R,\text{E-Let},\widehat{\beta}\}} \mathcal{P}_k[\![\sigma' e']\!]$.*

Proof: The proof is by induction on the size of σe and proceeds by case-analysis of the definition of $\to$. To prove that only R, E-Let, and $\widehat{\beta}$ are applied, we simply examine the reduction sequences below. Note that the auxilliary lemma 13 only employs $\widehat{\beta}$.

- Case E-Let.

$$
\begin{array}{rll}
 & S\ \mathcal{C}_k[\![E\langle \textsf{let (}x\textsf{ = }v\textsf{) }e\rangle]\!] & \\
= & S\ \textsf{let (}x\textsf{ = }\Phi[\![v]\!]\textsf{) }\mathcal{C}_k[\![E\langle e\rangle]\!] & \\
\to & S\ \mathcal{C}_k[\![E\langle e\rangle]\!][x/\Phi[\![v]\!]] & \text{E-Let} \\
= & S\ \mathcal{C}_k[\![E\langle e\rangle[x/v]]\!] & \text{lemma 14}
\end{array}
$$

  which is $\mathcal{C}_k$, applied to the the RHS of E-Let.

- Case E-Break.

$$
\begin{array}{rll}
 & S\ \mathcal{C}_k[\![E_1\langle label\textsf{:}T\ \textsf{\{ }E_2\langle \textsf{break }label\ v\rangle\ \textsf{\}}\rangle]\!] & \\
= & S\ \mathcal{K}_k[\![E_1]\!]\textsf{(}\Phi[\![v]\!]\textsf{)} & \\
\twoheadrightarrow & S\ \mathcal{C}_k[\![E_1\langle v\rangle]\!] & \text{lemma 13}
\end{array}
$$

- Case E-Label-Pop.

$$
\begin{array}{rl}
 & S\ \mathcal{C}_k[\![E\langle label\textsf{:}T\ \textsf{\{ }v\ \textsf{\}}\rangle]\!] \\
= & S\ \mathcal{C}_k[\![E\langle v\rangle]\!]
\end{array}
$$

- Case E-Ref.

$$\begin{array}{lll} & S\ \mathcal{C}_k[\![E\langle \textsf{ref}\ v\rangle]\!] & \\ = & S\ \textsf{let}\ x = \textsf{ref}\ \Phi[\![v]\!]\ \textsf{in}\ \mathcal{K}_k[\![E]\!](x) & \\ \rightarrow & S, (l, \Phi[\![v]\!])\ \textsf{let}\ x = l\ \textsf{in}\ \mathcal{K}_k[\![E]\!](x) & \text{E-Ref} \\ \rightarrow & S, (l, \Phi[\![v]\!])\ \mathcal{K}_k[\![E]\!](l) & \text{E-Let} \\ = & S, (l, \Phi[\![v]\!])\ \mathcal{K}_k[\![E]\!](\Phi(l)) & \\ \twoheadrightarrow & S, (l, \Phi[\![v]\!])\ \mathcal{C}_k[\![E\langle l\rangle]\!] & \text{lemma 13} \end{array}$$

- Case E-Deref.

$$\begin{array}{lll} & S\ \mathcal{C}_k[\![E\langle \textsf{deref}\ l\rangle]\!] & \\ = & S\ \textsf{let}\ x = \textsf{deref}\ l\ \textsf{in}\ \mathcal{K}_k[\![E]\!](x) & \\ \rightarrow & S\ \textsf{let}\ x = S(l)\ \textsf{in}\ \mathcal{K}_k[\![E]\!](x) & \text{E-Deref and } dom(S) = dom(\sigma) \\ \rightarrow & S\ \mathcal{K}_k[\![E]\!](S(l)) & \text{E-Let} \\ \twoheadrightarrow & S\ \mathcal{C}_k[\![E\langle v\rangle]\!] & \text{lemma 13, where } S(l) = \Phi(v) \end{array}$$

- Case E-SetRef.

$$\begin{array}{lll} & S\ \mathcal{C}_k[\![E\langle \textsf{setref}\ l\ v\rangle]\!] & \\ = & S\ \textsf{let}\ x = \textsf{setref}\ l\ \Phi[\![v]\!]\ \textsf{in}\ \mathcal{K}_k[\![E]\!](l) & \\ \rightarrow & (l, \Phi[\![v]\!])S\ \textsf{let}\ x = l\ \textsf{in}\ \mathcal{K}_k[\![E]\!](l) & \\ \rightarrow & (l, \Phi[\![v]\!])S\ \mathcal{K}_k[\![E]\!](l) & \text{since } x \text{ is unused} \\ \twoheadrightarrow & (l, \Phi[\![v]\!])S\ \mathcal{C}_k[\![E\langle l\rangle]\!] & \text{by lemma 13, since } \phi(l) = l \end{array}$$

- Case E-TagCheck.

$$\begin{array}{lll} & S\ \mathcal{C}_k[\![E\langle \textsf{tagcheck}\ R\ v\rangle]\!] & \\ = & S\ \textsf{let}\ x = \textsf{tagcheck}\ R\ \Phi[\![v]\!]\ \textsf{in}\ \mathcal{K}_k[\![E]\!](x) & \\ \rightarrow & S\ \mathcal{K}_k[\![E]\!](\Phi[\![v]\!]) & \text{E-TagCheck, by hypothesis and since } \textsf{typeof}(v) = \textsf{typeof}(\Phi[\![v]\!]) \\ \twoheadrightarrow & S\ \mathcal{C}_k[\![E\langle v\rangle]\!] & \text{lemma 13} \end{array}$$

■

Lemma 13 $S\mathcal{K}_k[\![E]\!](\Phi[\![v]\!]) \twoheadrightarrow_{\widehat{\beta}} S\mathcal{C}_k[\![E\langle v\rangle]\!]$.

Proof: The proof is by case analysis of E. We elide S in the steps below as it remains constant.

- Case $E\langle \textsf{ref}\ \bullet\rangle$.

$$\begin{array}{lll} & \mathcal{K}_k[\![E\langle \textsf{ref}\ \bullet\rangle]\!](\Phi[\![v]\!]) & \\ = & (\textsf{func}(x)\ \{\ \mathcal{C}_k[\![E\langle \textsf{ref}\ x\rangle]\!]\ \})(\Phi[\![v]\!]) & \\ \rightarrow & \mathcal{C}_k[\![E\langle \textsf{ref}\ x\rangle]\!][x/\Phi[\![v]\!]] & \widehat{\beta} \\ = & \mathcal{C}_k[\![E\langle \textsf{ref}\ x\rangle[x/v]]\!] & \text{by lemma 14} \\ = & \mathcal{C}_k[\![E\langle \textsf{ref}\ v\rangle]\!] & \end{array}$$

- Case $E\langle \textsf{break}\ label\ \bullet\rangle$.

$$\begin{array}{lll}
& \mathcal{K}_k[\![E\langle \textsf{break}\ label\ \bullet\rangle]\!](\Phi[\![v]\!]) & \\
= & (\textsf{func}(x)\ \{\ \mathcal{C}_k[\![E\langle \textsf{break}\ label\ x\rangle]\!]\ \})(\Phi[\![v]\!]) & \\
\to & \mathcal{C}_k[\![E\langle \textsf{break}\ label\ x\rangle]\!][x/\Phi[\![v]\!]] & \widehat{\beta} \\
= & \mathcal{C}_k[\![E\langle \textsf{break}\ label\ x\rangle[x/v]]\!] & \text{by lemma 14} \\
= & \mathcal{C}_k[\![E\langle \textsf{break}\ label\ v\rangle]\!] &
\end{array}$$

- Case $E\langle \textsf{if}\ (\bullet)\ \{\ M_1\ \}\ \textsf{else}\ \{\ M_2\ \}\rangle$.

$$\begin{array}{lll}
& \mathcal{K}_k[\![E\langle \textsf{if}\ (\bullet)\ \{\ e_2\ \}\ \textsf{else}\ \{\ e_3\ \}\rangle]\!](\Phi[\![v]\!]) & \\
= & \textsf{func}(x)\ \{\ \mathcal{C}_k[\![E\langle \textsf{if}\ (x)\ \{\ e_2\ \}\ \textsf{else}\ \{\ e_3\ \}\rangle]\!]\ \}(\Phi[\![v]\!]) & \\
\to & \mathcal{C}_k[\![E\langle \textsf{if}\ (x)\ \{\ e_2\ \}\ \textsf{else}\ \{\ e_3\ \}\rangle]\!][x/\Phi[\![v]\!]] & \widehat{\beta} \\
= & \mathcal{C}_k[\![E\langle \textsf{if}\ (x)\ \{\ e_2\ \}\ \textsf{else}\ \{\ e_3\ \}\rangle[x/v]]\!] & \text{lemma 14} \\
= & \mathcal{C}_k[\![E\langle \textsf{if}\ (v)\ \{\ e_2\ \}\ \textsf{else}\ \{\ e_3\ \}\rangle]\!] &
\end{array}$$

- Case $E\langle \bullet(e\cdots)\rangle$.

$$\begin{array}{lll}
& \mathcal{K}_k[\![E\langle \bullet(e\cdots)\rangle]\!](\Phi[\![v]\!]) & \\
= & \textsf{func}(f)\ \{\ \mathcal{C}_k[\![E\langle f(e\cdots)\rangle]\!]\ \}(\Phi[\![v]\!]) & \\
\to & \mathcal{C}_k[\![E\langle f(e\cdots)\rangle]\!][f/\Phi[\![v]\!]] & \widehat{\beta_v} \\
= & \mathcal{C}_k[\![E\langle v(e\cdots)\rangle]\!] & \text{lemma 14}
\end{array}$$

  Cases $E\langle v_f(v\cdots\bullet e\cdots)\rangle$ and $E\langle op_n(v\cdots\bullet e\cdots)\rangle$ are similar.

- Case $E\langle label\texttt{: \{ } \bullet \texttt{ \}}\rangle$.

$$\begin{array}{lll}
& \mathcal{K}_k[\![E\langle label\texttt{:\{ } \bullet \texttt{ \}}\rangle]\!](\Phi[\![v]\!]) & \\
= & \textsf{func}(x)\ \{\ \mathcal{C}_k[\![E\langle label\texttt{: \{ } x \texttt{ \} }\rangle]\!]\ \}(\Phi[\![v]\!]) & \\
\to & \mathcal{C}_k[\![E\langle label\texttt{: \{ } x \texttt{ \} }\rangle]\!](\Phi[\![v]\!]) & \widehat{\beta_v} \\
= & \mathcal{C}_k[\![E\langle label\texttt{: \{ } v \texttt{ \} }\rangle]\!] & \text{lemma 14}
\end{array}$$

- Case $E\langle \textsf{deref}\ \bullet\rangle$.

$$\begin{array}{lll}
& \mathcal{K}_k[\![E\langle \textsf{deref}\ \bullet\rangle]\!](\Phi[\![v]\!]) & \\
= & (\textsf{func}(x)\ \{\ \mathcal{C}_k[\![E\langle \textsf{deref}\ x\rangle]\!]\ \})(\Phi[\![v]\!]) & \\
\to & \mathcal{C}_k[\![E\langle \textsf{deref}\ x\rangle]\!][x/\Phi[\![v]\!]] & \widehat{\beta_v} \\
= & \mathcal{C}_k[\![E\langle \textsf{deref}\ x\rangle[x/v]]\!] & \text{by lemma 14} \\
= & \mathcal{C}_k[\![E\langle \textsf{deref}\ v\rangle]\!] &
\end{array}$$

- Case $E\langle \textsf{setref}\ \bullet\ e\rangle$.

$$\begin{array}{lll}
& \mathcal{K}_k[\![E\langle \textsf{setref}\ \bullet\ e\rangle]\!](\Phi[\![v]\!]) & \\
= & (\textsf{func}(x)\ \{\ \mathcal{C}_k[\![E\langle \textsf{setref}\ x\ e\rangle]\!]\ \})(\Phi[\![v]\!]) & \\
\to & \mathcal{C}_k[\![E\langle \textsf{setref}\ x\ e\rangle]\!][x/\Phi[\![v]\!]] & \widehat{\beta_v} \\
= & \mathcal{C}_k[\![E\langle \textsf{setref}\ x\ e\rangle[x/v]]\!] & \text{by lemma 14} \\
= & \mathcal{C}_k[\![E\langle \textsf{setref}\ v\ e\rangle]\!] &
\end{array}$$

Case $E\langle \textsf{setref}\ v\ \bullet\rangle$ is similar.

- Case $E\langle \textsf{tagcheck}\ R\ \bullet\rangle$.

$$\begin{array}{lll} & \mathcal{K}_k[\![E\langle \textsf{tagcheck}\ R\ \bullet\rangle]\!](\Phi[\![v]\!]) & \\ = & (\textsf{func}(x)\ \{\ \mathcal{C}_k[\![E\langle \textsf{tagcheck}\ R\ x\rangle]\!]\ \})(\Phi[\![v]\!]) & \\ \rightarrow & \mathcal{C}_k[\![E\langle \textsf{tagcheck}\ R\ x\rangle]\!][x/\Phi[\![v]\!]] & \widehat{\beta_v} \\ = & \mathcal{C}_k[\![E\langle \textsf{tagcheck}\ R\ x\rangle[x/v]]\!] & \text{by lemma 14} \\ = & \mathcal{C}_k[\![E\langle \textsf{tagcheck}\ R\ v\rangle]\!] & \end{array}$$

■

Lemma 14 (Substitution Commutes with $\mathcal{C}_k$) $\mathcal{C}_k[\![e]\!][x/\Phi(v)] = \mathcal{C}_k[\![e[x/v]]\!]$.

Proof: by induction on the structure of e. For each inductive case of $\mathcal{C}_k$, expand the definitions of Φ and $\mathcal{K}_k$ on the RHS to get a term with $\mathcal{C}_k$ applied to subterms of e. The inductive hypotheses now trivially apply. ■

1.4 Additional Rules for Flow Analysis

For soundness of flow analysis, we need to reason about intermediate terms with concrete locations, l. Hence, we extend abstract heaps to the type:

$\boxed{\widehat{S} : \hat{l} + l \rightarrow R}$

$\boxed{\widehat{S}; \widehat{\Gamma} \triangleright V \rightsquigarrow \widehat{V}}$

$$\frac{\widehat{S}(\hat{l}) = R \qquad \mathit{runtime}(V) = R}{\widehat{S}; \widehat{\Gamma} \triangleright V \rightsquigarrow \mathsf{Deref}\ \hat{l}\ R} \tag{V-Deref}$$

$$\frac{\widehat{S}; \widehat{\Gamma} \triangleright V \rightsquigarrow \mathsf{Deref}\ \hat{l}}{\widehat{S}; \widehat{\Gamma} \triangleright \delta_1(\textsf{typeof}, V) \rightsquigarrow \mathsf{LocTypeof}\ \hat{l}} \tag{V-Typeof}$$

$$\frac{\widehat{S}; \widehat{\Gamma} \triangleright V_1 \rightsquigarrow \mathsf{LocType}\ \hat{l} \qquad V_2 \in \widehat{S}(\hat{l})}{\widehat{S}; \widehat{\Gamma} \triangleright \delta_2(\texttt{===}, V_1, V_2) \rightsquigarrow \mathsf{LocType}\ \hat{l}\ R} \tag{V-TypeIs}$$

$\boxed{\widehat{S}; \widehat{\Gamma} \vDash SM}$

$$\frac{\widehat{S}; \widehat{\Gamma} \triangleright V \rightsquigarrow \widehat{V} \qquad \widehat{S}; x : r, \widehat{\Gamma} \vDash S\ M}{\widehat{S}; \widehat{\Gamma} \vDash S\ \textsf{let}\ (x\ \texttt{=}\ \textsf{deref}\ V)\ M} \tag{F-Deref-Unk}$$

$$\frac{\widehat{S}; \widehat{\Gamma} \triangleright V \rightsquigarrow \widehat{V} \qquad \widehat{S}; x : \{\texttt{"string"}\}, \widehat{\Gamma} \vDash S\ M}{\widehat{S}; \widehat{\Gamma} \vDash S\ \textsf{let}\ (x\ \texttt{=}\ \textsf{typeof}\ V)\ M} \tag{F-Typeof-Unk}$$

$$\frac{\widehat{S};\widehat{\Gamma} \triangleright V_1 \rightsquigarrow \widehat{V_1} \qquad \widehat{S};\widehat{\Gamma} \triangleright V_2 \rightsquigarrow \widehat{V_2} \qquad \widehat{S}; x : \{\texttt{"boolean"}\}, \widehat{\Gamma} \vDash S\ M}{\widehat{S};\widehat{\Gamma} \vDash S\ \textsf{let}\ (x\ \texttt{=}\ V_1\ \texttt{===}\ V_2)\ M} \quad \text{(F-Eq)}$$

$$\frac{\widehat{S};\widehat{\Gamma} \triangleright V_1 \rightsquigarrow \textsf{Ref}\ \hat{l}_1 \qquad \widehat{S};\widehat{\Gamma} \triangleright V_2 \rightsquigarrow \textsf{Ref}\ \hat{l}_2 \qquad \hat{l} : R, \widehat{S}; x : r, del(\hat{l}_1, del(\hat{l}_2, \widehat{\Gamma})) \vDash S\ M}{\widehat{S};\widehat{\Gamma} \vDash S\ \textsf{let}\ (x\ \texttt{=}\ \textsf{setref}\ V_1\ V_2)\ M} \quad \text{(F-SetRef-Alias1)}$$

$$\frac{\widehat{S};\widehat{\Gamma} \triangleright V_1 \rightsquigarrow \widehat{V_1} \qquad \widehat{S};\widehat{\Gamma} \triangleright V_2 \rightsquigarrow \textsf{Ref}\ \hat{l}_2 \qquad \hat{l} : R, \widehat{S}; x : r, del(\hat{l}_2, \widehat{\Gamma}) \vDash S\ M}{\widehat{S};\widehat{\Gamma} \vDash S\ \textsf{let}\ (x\ \texttt{=}\ \textsf{setref}\ V_1\ V_2)\ M} \quad \text{(F-SetRef-Alias2)}$$

$$\frac{\widehat{S};\widehat{\Gamma} \triangleright V \rightsquigarrow \widehat{V} \qquad \widehat{S};\widehat{\Gamma} \vDash M_1 \qquad \widehat{S};\widehat{\Gamma} \vDash M_2}{\widehat{S};\widehat{\Gamma} \vDash S\ \textsf{if}\ (V)\ \{\ M_1\ \}\ \textsf{else}\ \{\ M_2\ \}} \quad \text{(F-If)}$$

$$\frac{\widehat{S};\widehat{\Gamma} \vDash M_1}{\widehat{S};\widehat{\Gamma} \vDash S\ \textsf{if}\ (\textsf{true})\ \{\ M_1\ \}\ \textsf{else}\ \{\ M_2\ \}} \quad \text{(F-If-True)}$$

$$\frac{\widehat{S};\widehat{\Gamma} \vDash M_2}{\widehat{S};\widehat{\Gamma} \vDash S\ \textsf{if}\ (\textsf{false})\ \{\ M_1\ \}\ \textsf{else}\ \{\ M_2\ \}} \quad \text{(F-If-False)}$$

1.5 Flow Analysis

Furthermore, flow analysis is sound:

Lemma 15 (Soundness of Flow Analysis) *If $\hat{S}, \cdot \vDash SM$ and $SM \to S'M'$, then either:*

- *$\hat{S}', \cdot \vDash S'M'$, or*
- *M is a $\widehat{\beta_v}$ redex, $\textsf{func}(x \cdots)\ :\ T \cdots \bot\ \{\ N\ \}(V \cdots)$, where for some V, $\delta_1(\textsf{typeof}, V) \notin runtime(T)$.*

Proof: by case-analysis on M.

- Case $S\ \textsf{func}(x \cdots)\ :\ T \cdots \bot\ \{\ N\ \}(V \cdots) \to SN[x/V] \cdots$.

  By inversion, $\cdot; x : runtime(T) \cdots \vDash N$ (V-Restart) and there exist $R \cdots$, such that $\cdot \triangleright V \rightsquigarrow R \cdots$. Since $V \cdots$ are all closed, $\delta_1(\textsf{typeof}, V) \cdots$ is defined. By case analysis on V, $\cdot;\cdot \triangleright V \rightsquigarrow \{\delta_1(\textsf{typeof}, V)\} \cdots$. There are two cases:

  - If $\{\delta_1(\textsf{typeof}, V)\} \sqsubseteq runtime(T) \cdots$, then by substitution (lemma 16), $\cdot;\cdot \vDash N[x/V \cdots]$.
  - Otherwise, for some V, T, $\delta_1(\textsf{typeof}, V) \notin runtime(T)$.

- Case S **let** x **=** V**in** $M \to SM[x/V]$.

  By F-LetVal, $\widehat{S}; \cdot \vDash S$ **let** x **=** V**in** M. By hypothesis, $\widehat{S}; x : \widehat{V} \vDash SM$ and $\widehat{S}; \cdot \triangleright V \rightsquigarrow \widehat{V}$. By substitution (lemma 16), $\widehat{S}; \cdot \vDash SM[x/V]$.

- Case S **let** x **=** **ref** V**in** $M \to (l, V)S$ **let** x **=** l**in** M.

  There are two applicable rules.

  - By F-Alloc, $\widehat{S}; \cdot \triangleright V \rightsquigarrow R$ and $\hat{l} : R, \widehat{S}; x : \mathsf{Ref}\ \hat{l} \vDash SM$. By V-Loc, $(l, V), \widehat{S}; \cdot \triangleright l \rightsquigarrow \mathsf{Ref}\ l$. Renaming $\hat{l}$ to l (lemma 17), $l : R, \widehat{S}; x : \mathsf{Ref}\ l \vDash SM$. Hence by F-LetVal, $l : R; \widehat{S}; \cdot \vDash (l, V), S$ **let** x **=** l**in** M.
  - By F-Ref-Alias, $\widehat{S}; \cdot \triangleright V \rightsquigarrow \mathsf{Ref}\ \hat{l}$ and $\widehat{S}; x : r \vDash SM$. By V-Loc, followed by V-Sub, $(l, v), \widehat{S}; \cdot \triangleright \hat{l} \rightsquigarrow r$. Hence by F-LetVal, $l : r, \widehat{S}; \cdot \vDash (l, V), S$ **let** x **=** l**in** M.

- Case S **let** x **=** **deref** l **in** $M \to S$ **let** x **=** $S(l)$ **in** M.

  - By F-Deref, $\widehat{S}; \cdot \triangleright l \rightsquigarrow \mathsf{Ref}\ l$, $\widehat{S}(l) = R$, and $\widehat{S}; x : \mathsf{Deref}\ l\ R \vDash S\ M$. By V-Deref, $\widehat{S}; \cdot \triangleright S(l) \rightsquigarrow \mathsf{Deref}\ l\ R$. Apply F-LetVal to the RHS.
  - By F-Deref-Unk, $\widehat{S}; x : r \vDash SM$ and $\widehat{S}; \cdot \triangleright S(l) \rightsquigarrow \widehat{V}$. By V-Sub and F-LetVal, the conclusion follows.

- Case S **let** x **=** **typeof** V **in** $M \to S$ **let** x **=** $\delta_1($**typeof**$, V)$ **in** M

  - By F-Typeof, $\cdot \triangleright V \rightsquigarrow \mathsf{Deref}\ \hat{l}\ R$ and $\widehat{S}; x : \mathsf{LocTypeof}\ \hat{l} \vDash M$. By V-Typeof, $\widehat{S}; \cdot \triangleright \delta_1($**typeof**$, V) \rightsquigarrow \mathsf{LocTypeof}\ \hat{l}$. Apply F-LetVal to the RHS.
  - By F-Typeof-Unk, $\widehat{S}; x : \{\texttt{"string"}\} \vDash SM$ and $\widehat{S}; \cdot \triangleright V \rightsquigarrow \widehat{V}$. By definition of $\delta_1($**typeof**$)$, $\widehat{S}; \cdot \triangleright \delta_1($**typeof**$, V) \rightsquigarrow \{\texttt{"string"}\}$. Apply F-LetVal.

- Case S **let** x **=** $(V_1$ **===** $V_2)$ **in** $M \to S\ S$ **let** x **=** $\delta_2($**===**$, V_1, V_2)$ **in** M.

  - By F-TypeIs-Str, $V_2 = \texttt{"string"}$, $\widehat{S}; \cdot \triangleright V_1 \rightsquigarrow \mathsf{LocTypeof}\ \hat{l}$, and $\widehat{S}; x : \mathsf{LocType}\ \hat{l}\ \{\texttt{"string"}\} \vDash M$. By V-TypeIs, $\widehat{S}; \cdot \triangleright \delta_2($**===**$, V_1, V_2) \rightsquigarrow \mathsf{LocTypeof}\ \hat{l}$. Apply F-LetVal to the RHS.
  - By F-Eq, $\widehat{S}; x : \{\texttt{"boolean"}\} \vDash SM$, $\widehat{S}; \cdot \triangleright V_1 \rightsquigarrow \widehat{V_1}$, and $\widehat{S}; \cdot \triangleright V_2 \rightsquigarrow \widehat{V_2}$. By definition of $\delta_2($**===**$)$, $\widehat{S}; \cdot \triangleright \delta_2($**===**$, V_1, V_2) \rightsquigarrow \{\texttt{"boolean"}\}$. Apply F-LetVal.

- Case S **let** x **=** **tagcheck** $R\ V$ **in** $M \to S$ **let** x **=** V **in** M.

  By F-TagCheck, $\widehat{S}; \cdot \triangleright V \rightsquigarrow R$. Apply F-LetVal to the RHS.

- Case S **let** x **=** **setref** $l\ V$ **in** M.

  - By F-SetRef, $\widehat{S}; \cdot \triangleright l \rightsquigarrow \mathsf{Ref}\ l$, $\widehat{S}; \cdot \triangleright V \rightsquigarrow R$, and $\widehat{S}[l := R], x : \mathsf{Ref}\ l \vDash SM$. Apply F-LetVal.
  - By F-SetRef-Alias2, $\widehat{S}; \cdot \triangleright l \rightsquigarrow \mathsf{Ref}\ l$, $\widehat{S}; \cdot \triangleright V \rightsquigarrow \mathsf{Ref}\ \hat{l}'$, and $\widehat{S}; x : r \vDash SM$. Apply F-LetVal.

- Case S **if** **(true)** **{** M_1 **}** **else** **{** M_2 **}** $\to S\ M_1$.

  By F-If-True, $\widehat{S}; \cdot \triangleright V \rightsquigarrow \mathsf{LocType}\ \hat{l}\ R$, $R \sqsubseteq \widehat{S}(\hat{l})$, and $\widehat{S}; \cdot \vDash M_1$.

- Case S **if (false)** { M_1 } **else** { M_2 } $\rightarrow S\ M_2$.

  By F-If-False, $\widehat{S};\cdot \triangleright V \rightsquigarrow \mathsf{LocType}\ \hat{l}\ R$, $R \sqsubseteq \widehat{S}(\hat{l})$, and $\widehat{S};\cdot \vDash M_2$.

Lemma 16 (Substitution) *If* $\widehat{S};x : \widehat{V},\widehat{\Gamma} \vDash M$ *and* $\widehat{S};\widehat{\Gamma} \triangleright V \rightsquigarrow \widehat{V'}$, *with* $\widehat{V'} \sqsubseteq \widehat{V}$ *then* $\widehat{S};\widehat{\Gamma} \vDash M[x/V']$.

Proof by induction on $\widehat{S};x : \widehat{V},\widehat{\Gamma} \vDash M$.

The key to this proof is substituting values W that occur in expressions M. i.e., we must show that if $\widehat{S};x : \widehat{V},\widehat{\Gamma} \triangleright W \rightsquigarrow \widehat{W}$, then $\widehat{S};\widehat{\Gamma} \triangleright W[x/V'] \rightsquigarrow \widehat{W}$.

The interesting case is when $W = \textbf{func}(y\cdots):T\cdots \rightarrow \bot$ { M }. So, by V-Restart:

$$\begin{array}{rl} & \cdot; y : runtime(T)\cdots, reset(x : \widehat{V},\widehat{\Gamma}) \vDash M' \\ = & \cdot; reset(x : \widehat{V''}), y : runtime(T)\cdots, reset(\widehat{\Gamma}) \vDash M' \end{array}$$

By definition of *reset*, $\widehat{V} \sqsubseteq \widehat{V''}$. Since M' is smaller than M, the inductive hypothesis applies.

Lemma 17 (Renaming Locations) *If* $\widehat{l} : \widehat{V},\widehat{S};\widehat{\Gamma} \vDash SM$ *and* $l \notin dom(S)$, *then* $l : \widehat{V},\widehat{S};\widehat{\Gamma}[\widehat{l}/l] \vDash SM$.

Proof by induction on the size of $\widehat{\Gamma}$. ■

1.6 Combined Soundness Theorems

Theorem 1 (Strengthened Progress) *If:*

i. $\Sigma;\cdot \vdash e : T$,

ii. $\Sigma \vdash \sigma$, *and*

iii. $\widehat{S};\cdot \vDash \mathcal{P}_k[\![\sigma e]\!]$,

then either:

i. $e \in v$, *or*

ii. There exist σ' *and* e', *such that* $\sigma e \rightarrow \sigma' e'$.

Proof: This follows from lemma 1, with the possibility of **tagerr**s eliminated by inspection of the acceptability relation—flow analysis does not admit expressions with **tagerr**s. ■

Theorem 2 (Combined Preservation) *If:*

i. $\Sigma;\cdot \vdash e : T$,

ii. $\Sigma \vdash \sigma$,

ii. $\widehat{S};\cdot \vDash \mathcal{P}_k[\![\sigma e]\!]$, *and*

iv. $\sigma e \to \sigma' e'$,

then there exist Σ' *and* $\widehat{S}'$, *such that:*

i. $\Sigma'; \cdot \vdash e' : T$,

ii. $\Sigma' \vdash \sigma'$,

iii. $\Sigma \subseteq \Sigma'$, *and*

iv. $\hat{S}'; \cdot \vDash \mathcal{P}_k[\![\sigma' e']\!]$.

Proof: Conclusions (i.), (ii.), and (iii.) follow immediately from lemma 2. For conclusion (iv.), apply lemma 12 to hypothesis (iv.) to get a reduction sequence, $\mathcal{P}_k[\![\sigma e]\!] \twoheadrightarrow \mathcal{P}_k[\![\sigma' e']\!]$. Apply lemma 15 at each step, eliminating case (ii.) of the lemma as follows. By lemma 12, intermediate expressions are not β_v-redexes, so case (ii.) does not apply. Suppose e itself has an active β_v-redex:

$$e = E\langle \textsf{func}(x \cdots) \ : \ U \cdots \to S \ \{ \ e_f \ \}(v \cdots) \rangle$$

Transform e to CPS:

$$\mathcal{C}_k[\![e]\!] \quad = \quad \Phi[\![\textsf{func}(x \cdots) \ : \ U \cdots \to S \ \{ \ e_f \ \}]\!](\mathcal{K}_k[\![E]\!], \Phi[\![v]\!] \cdots)$$

Since e is typable, there exists a Γ such that:

$$\Sigma; \Gamma \vdash \textsf{func}(x \cdots) \ : \ U \cdots \to S \ \{ \ e_f \ \}(v \cdots) : S$$

Furthermore, by inversion (lemma 8), $\Sigma; \Gamma \vdash v \ : \ U \cdots$. For all v, $\delta_1(\textsf{typeof}, v) \in \mathit{runtime}(U)$ (lemma 11). By inspection of Φ, $\delta_1(\textsf{typeof}, v) = \delta_1(\textsf{typeof}, \Phi[\![v]\!])$, case (ii.) of lemma 15 does not apply. ■

Bibliography

[1] A. Sabry and M. Felleisen. Reasoning about programs in continuation-passing style. *LISP and Symbolic Computation*, 6(3), 1993.