A Simple Abstract Interpretation Framework for Modular Analysis

JOONHYUP LEE and KWANGKEUN YI

1 SYNTAX AND SEMANTICS

1.1 Abstract Syntax

$$
\begin{array}{rcll}
\text{Identifiers} & x & \in & \text{Var} \\
\text{Expression} & e & \to & x \mid \lambda x.e \mid e\ e & \lambda\text{-calculus} \\
& & \mid & e \rtimes e & \text{linked expression} \\
& & \mid & \varepsilon & \text{empty module} \\
& & \mid & x = e\ ;\ e & \text{(recursive) binding}
\end{array}
$$

Fig. 1. Abstract syntax of the language.

1.2 Operational Semantics

$$
\begin{array}{rcll}
\text{Environment} & \sigma & \in & \text{Env} \\
\text{Location} & \ell & \in & \text{Loc} \triangleq \{\text{infinite set of locations}\} \\
\text{Value} & v & \in & \text{Val} \triangleq \text{Env} + \text{Var} \times \text{Expr} \times \text{Env} \\
\text{Weak Value} & w & \in & \text{WVal} \triangleq \text{Val} + \text{Loc} \times \text{Val} \\
\text{Environment} & \sigma & \to & \bullet & \text{empty stack} \\
& & \mid & (x, \ell) :: \sigma & \text{free location binding} \\
& & \mid & (x, w) :: \sigma & \text{weak value binding} \\
\text{Value} & v & \to & \sigma & \text{exported environment} \\
& & \mid & \langle \lambda x.e, \sigma \rangle & \text{closure} \\
\text{Weak Value} & w & \to & v & \text{value} \\
& & \mid & \mu\ell.v & \text{recursive value}
\end{array}
$$

Fig. 2. Definition of the semantic domains.

1.3 Reconciling with Conventional Backpatching

The semantics in Figure 3 makes sense due to similarity with a conventional backpatching semantics as presented in Figure 5. We have defined a relation $\sim$ that satisfies:

$$\sim \subseteq \text{WVal} \times (\text{MVal} \times \text{Mem} \times \mathscr{P}(\text{Loc})) \qquad \bullet \sim (\bullet, \varnothing, \varnothing)$$

Authors' address: Joonhyup Lee; Kwangkeun Yi.

XXXX-XXXX/2024/3-ART $15.00

https://doi.org/10.1145/nnnnnnn.nnnnnnn

$$\boxed{\sigma \vdash e \Downarrow v}$$

$$\textsc{Id}\ \frac{\sigma(x) = v}{\sigma \vdash x \Downarrow v} \qquad \textsc{RecId}\ \frac{\sigma(x) = \mu\ell.v}{\sigma \vdash x \Downarrow v[\mu\ell.v/\ell]} \qquad \textsc{Fn}\ \frac{}{\sigma \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle} \qquad \textsc{App}\ \frac{\sigma \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle \quad \sigma \vdash e_2 \Downarrow v_2 \quad (x, v_2) :: \sigma_1 \vdash e \Downarrow v}{\sigma \vdash e_1\ e_2 \Downarrow v}$$

$$\textsc{Link}\ \frac{\sigma \vdash e_1 \Downarrow \sigma_1 \quad \sigma_1 \vdash e_2 \Downarrow v}{\sigma \vdash e_1 \rtimes e_2 \Downarrow v} \qquad \textsc{Empty}\ \frac{}{\sigma \vdash \varepsilon \Downarrow \bullet} \qquad \textsc{Bind}\ \frac{\ell \notin \mathrm{FLoc}(\sigma) \quad (x, \ell) :: \sigma \vdash e_1 \Downarrow v_1 \quad (x, \mu\ell.v_1) :: \sigma \vdash e_1 \Downarrow \sigma_2}{\sigma \vdash x = e_1; e_2 \Downarrow (x, \mu\ell.v_1) :: \sigma_2}$$

Fig. 3. The big-step operational semantics.

$$\begin{array}{rclll}
\text{Environment} & \sigma & \in & \mathrm{MEnv} \triangleq \mathrm{Var} \xrightarrow{\text{fin}} \mathrm{Loc} & \\
\text{Memory} & m & \in & \mathrm{Mem} \triangleq \mathrm{Loc} \xrightarrow{\text{fin}} \mathrm{MVal} & \\
\text{Allocated set} & L & \subseteq & \mathrm{Loc} & \\
\text{Value} & v & \in & \mathrm{MVal} \triangleq \mathrm{MEnv} + \mathrm{Var} \times \mathrm{Expr} \times \mathrm{MEnv} & \\
\text{Environment} & \sigma & \rightarrow & \bullet & \text{empty stack} \\
 & & \mid & (x, \ell) :: \sigma & \text{location binding} \\
\text{Value} & v & \rightarrow & \sigma & \text{exported environment} \\
 & & \mid & \langle \lambda x.e, \sigma \rangle & \text{closure}
\end{array}$$

Fig. 4. Definition of the semantic domains with memory.

and the following theorem:

Theorem 1.1 (Equivalence of semantics). For all $\sigma \in \mathrm{Env}, \sigma' \in \mathrm{MEnv} \times \mathrm{Mem} \times \mathscr{P}(\mathrm{Loc}), v \in \mathrm{Val}, v' \in \mathrm{MVal} \times \mathrm{Mem} \times \mathscr{P}(\mathrm{Loc})$, we have:

$$\sigma \sim \sigma' \text{ and } \sigma \vdash e \Downarrow v \Rightarrow \exists v' : v \sim v' \text{ and } \sigma' \vdash e \Downarrow v'$$
$$\sigma \sim \sigma' \text{ and } \sigma' \vdash e \Downarrow v' \Rightarrow \exists v : v \sim v' \text{ and } \sigma \vdash e \Downarrow v$$

The actual definition for $\sim$ can be found in the appendix.

$$\boxed{\sigma, m, L \vdash e \Downarrow v, m', L'}$$

$$\textsc{Id}\ \frac{\sigma(x) = \ell \qquad m(\ell) = v}{\sigma, m, L \vdash x \Downarrow v, m, L} \qquad \textsc{Fn}\ \frac{}{\sigma, m, L \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle, m, L}$$

$$\textsc{App}\ \frac{\begin{array}{c}\sigma, m, L \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle, m_1, L_1 \qquad \sigma, m_1, L_1 \vdash e_2 \Downarrow v_2, m_2, L_2 \qquad \ell \notin \mathrm{dom}(m_2) \cup L_2 \\ (x, \ell) :: \sigma_1, m_2[\ell \mapsto v_2], L_2 \vdash e \Downarrow v, m', L'\end{array}}{\sigma, m, L \vdash e_1\ e_2 \Downarrow v, m', L'}$$

$$\textsc{Link}\ \frac{\sigma, m, L \vdash e_1 \Downarrow \sigma_1, m_1, L_1 \qquad \sigma_1, m_1, L_1 \vdash e_2 \Downarrow v, m', L'}{\sigma, m, L \vdash e_1 \rtimes e_2 \Downarrow v, m', L'} \qquad \textsc{Empty}\ \frac{}{\sigma, m, L \vdash \varepsilon \Downarrow \bullet, m, L}$$

$$\textsc{Bind}\ \frac{\begin{array}{c}\ell \notin \mathrm{dom}(m) \cup L \qquad (x, \ell) :: \sigma, m, L \cup \{\ell\} \vdash e_1 \Downarrow v_1, m_1, L_1 \\ (x, \ell) :: \sigma, m_1[\ell \mapsto v_1], L_1 \vdash e_2 \Downarrow \sigma_2, m', L'\end{array}}{\sigma, m, L \vdash x = e_1; e_2 \Downarrow (x, \ell) :: \sigma_2, m', L'}$$

Fig. 5. The big-step operational semantics with memory.

2 GENERATING AND RESOLVING EVENTS

Now we formulate the semantics for generating events.

$$
\begin{array}{rrcll}
\text{Event} & E & \rightarrow & \mathsf{Init} & \text{initial environment} \\
 & & \mid & \mathsf{Read}(E, x) & \text{read event} \\
 & & \mid & \mathsf{Call}(E, v) & \text{call event} \\
\text{Environment} & \sigma & \rightarrow & \cdots & \\
 & & \mid & [E] & \text{answer to an event} \\
\text{Value} & v & \rightarrow & \cdots & \\
 & & \mid & E & \text{answer to an event}
\end{array}
$$

Fig. 6. Definition of the semantic domains with events. All other semantic domains are equal to Figure 2.

We extend how to read weak values given an environment.

$$
\begin{array}{rclrcl}
\bullet(x) & \triangleq & \bot & ((x', \ell) :: \sigma)(x) & \triangleq & (x = x' ? \ell : \sigma(x)) \\
[E](x) & \triangleq & \mathsf{Read}(E, x) & ((x', w) :: \sigma)(x) & \triangleq & (x = x' ? w : \sigma(x))
\end{array}
$$

Then we need to add only one rule to the semantics in Figure 3 for the semantics to incorporate events.

$$
\textsc{AppEvent}\quad \frac{\sigma \vdash e_1 \Downarrow E \qquad \sigma \vdash e_2 \Downarrow v}{\sigma \vdash e_1\ e_2 \Downarrow \mathsf{Call}(E, v)}
$$

Now we need to formulate the *concrete linking* rules. The concrete linking rule $\sigma_0 \infty w$, given an answer σ_0 to the Init event, resolves all events within w to obtain a set of final results.

Concrete linking makes sense because of the following theorem. First define:

$$
\text{eval}(e, \sigma) \triangleq \{v | \sigma \vdash e \Downarrow v\} \qquad \text{eval}(e, \Sigma) \triangleq \bigcup_{\sigma \in \Sigma} \text{eval}(e, \sigma) \qquad \sigma_0 \infty W \triangleq \bigcup_{w \in W} (\sigma_0 \infty w)
$$

Then the following holds:

Theorem 2.1 (Soundness of concrete linking). Given $e \in \text{Expr}, \sigma \in \text{Env}, v \in \text{Val}$,

$$
\forall \sigma_0 \in \text{Env} : \text{eval}(e, \sigma_0 \infty \sigma) \subseteq \sigma_0 \infty \text{eval}(e, \sigma)
$$

$$\boxed{\infty \in \mathrm{Env} \to \mathrm{Event} \to \mathscr{P}(\mathrm{Val})}$$

$$\begin{aligned}
\sigma_0 \infty \mathsf{Init} &\triangleq \{\sigma_0\} \\
\sigma_0 \infty \mathsf{Read}(E, x) &\triangleq \{v_+ | \sigma_+ \in \sigma_0 \infty E \wedge \sigma_+(x) = v_+\} \\
&\cup \{v_+[\mu\ell.v_+/\ell] | \sigma_+ \in \sigma_0 \infty E \wedge \sigma_+(x) = \mu\ell.v_+\} \\
\sigma_0 \infty \mathsf{Call}(E, v) &\triangleq \{v_+' | \langle \lambda x.e, \sigma_+\rangle \in \sigma_0 \infty E \wedge v_+ \in \sigma_0 \infty v \wedge (x, v_+) :: \sigma_+ \vdash e \Downarrow v_+'\} \\
&\cup \{\mathsf{Call}(E_+, v_+) | E_+ \in \sigma_0 \infty E \wedge v_+ \in \sigma_0 \infty v\}
\end{aligned}$$

$$\boxed{\infty \in \mathrm{Env} \to \mathrm{Env} \to \mathscr{P}(\mathrm{Env})}$$

$$\begin{aligned}
\sigma_0 \infty \bullet &\triangleq \{\bullet\} \\
\sigma_0 \infty (x, \ell) :: \sigma &\triangleq \{(x, \ell) :: \sigma_+ | \sigma_+ \in \sigma_0 \infty \sigma\} \\
\sigma_0 \infty (x, w) :: \sigma &\triangleq \{(x, w_+) :: \sigma_+ | w_+ \in \sigma_0 \infty w \wedge \sigma_+ \in \sigma_0 \infty \sigma\} \\
\sigma_0 \infty [E] &\triangleq \{\sigma_+ | \sigma_+ \in \sigma_0 \infty E\} \cup \{[E_+] | E_+ \in \sigma_0 \infty E\}
\end{aligned}$$

$$\boxed{\infty \in \mathrm{Env} \to \mathrm{Val} \to \mathscr{P}(\mathrm{Val})}$$

$$\sigma_0 \infty \langle \lambda x.e, \sigma\rangle \triangleq \{\langle \lambda x.e, \sigma_+\rangle | \sigma_+ \in \sigma_0 \infty \sigma\}$$

$$\boxed{\infty \in \mathrm{Env} \to \mathrm{WVal} \to \mathscr{P}(\mathrm{WVal})}$$

$$\sigma_0 \infty \mu\ell.v \triangleq \{\mu\ell.v_+ | v_+ \in \sigma_0 \infty v\}$$

Fig. 7. Definition for concrete linking.

3 TYPING

The definitions for types are in Figure 8 and the typing rules are in Figure 9.

$$
\begin{array}{rcclL}
\text{Type} & \tau & \in & \text{Type} & \\
\text{Type Environment} & \Gamma & \in & \text{TyEnv} & \\
\text{Abstract Event} & E^\# & \in & \text{Event}^\# & \\
\text{Safe Set} & S & \subseteq & \text{Var} & \\
\text{Abstract Event} & E^\# & \rightarrow & \mathsf{Init}^\# & \text{initial environment} \\
 & & | & \mathsf{Read}^\#(E^\#, x) & \text{read event} \\
 & & | & \mathsf{Call}^\#(E^\#, \tau) & \text{call event} \\
\text{Type Environment} & \Gamma & \rightarrow & \bullet & \text{empty environment} \\
 & & | & (x, \tau) :: \Gamma & \text{type binding} \\
 & & | & [E^\#] & \text{abstract event} \\
\text{Type} & \tau & \rightarrow & \Gamma & \text{module type} \\
 & & | & \tau \xrightarrow{S} \tau & \text{function type} \\
 & & | & E^\# & \text{abstract event}
\end{array}
$$

Fig. 8. Definition of types.

$$\boxed{\Gamma \vdash e : \tau, S}$$

$$
\text{T-ID}\ \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau, \{x\}}
\qquad
\text{T-FN}\ \frac{(x, \tau_1) :: \Gamma \vdash e : \tau_2, S}{\Gamma \vdash \lambda x.e : \tau_1 \xrightarrow{S} \tau_2, \varnothing}
\qquad
\text{T-APP}\ \frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{S} \tau_2, S_1 \qquad \Gamma \vdash e_2 : \tau_1, S_2}{\Gamma \vdash e_1\ e_2 : \tau_2, S_1 \cup S_2 \cup S}
$$

$$
\text{T-APPEVENT}\ \frac{\Gamma \vdash e_1 : E^\#, S_1 \qquad \Gamma \vdash e_2 : \tau, S_2}{\Gamma \vdash e_1\ e_2 : \mathsf{Call}^\#(E^\#, \tau), S_1 \cup S_2}
\qquad
\text{T-LINK}\ \frac{\Gamma \vdash e_1 : \Gamma_1, S_1 \qquad \Gamma_1 \vdash e_2 : \tau_2, S_2}{\Gamma \vdash e_1 \rtimes e_2 : \tau_2, S_1 \cup S_2}
\qquad
\text{T-NIL}\ \frac{}{\Gamma \vdash \varepsilon : \bullet, \varnothing}
$$

$$
\text{T-BIND}\ \frac{(x, \tau_1) :: \Gamma \vdash e_1 : \tau_1, S_1 \qquad x \notin S_1 \qquad (x, \tau_1) :: \Gamma \vdash e_2 : \Gamma_2, S_2}{\Gamma \vdash x = e_1; e_2 : (x, \tau_1) :: \Gamma_2, S_1 \cup S_2}
$$

Fig. 9. The typing judgment.

3.1 Type Safety

THEOREM 3.1 (TYPE SAFETY). For all $e \in \text{Expr}$, if $\bullet \vdash e : \tau, S$ for some τ, S, then e runs under $\bullet$ without error.

SKETCH. We prove this through unary logical relations and induction on the typing judgment.

Event Relation $\boxed{\mathcal{E}[\![E^\#]\!]\rho}$

$$\begin{aligned}
\mathcal{E}[\![\mathsf{Init}^\#]\!]\rho &\triangleq \{\mathsf{Init}\}\\
\mathcal{E}[\![\mathsf{Read}^\#(E^\#, x)]\!]\rho &\triangleq \{\mathsf{Read}(E, x) | E \in \mathcal{E}[\![E^\#]\!]\rho\}\\
\mathcal{E}[\![\mathsf{Call}^\#(E^\#, \tau)]\!]\rho &\triangleq \{\mathsf{Call}(E, v) | E \in \mathcal{E}[\![E^\#]\!]\rho \land v \in \mathcal{V}[\![\tau]\!]\rho\}
\end{aligned}$$

Context Relation $\boxed{C[\![\Gamma]\!]\rho}$

$$\begin{aligned}
C[\![\bullet]\!]\rho &\triangleq \{\bullet\}\\
C[\![(x, \tau) :: \Gamma]\!]\rho &\triangleq \{(x, w) :: \sigma | w \in \mathcal{W}[\![\tau]\!]\rho \land \sigma \in C[\![\Gamma]\!]\rho\}\\
&\cup \{(x, \ell) :: \sigma | \rho(\ell) = \tau \land \sigma \in C[\![\Gamma]\!]\rho\}\\
C[\![[E^\#]]\!]\rho &\triangleq \{[E] | E \in \mathcal{E}[\![E^\#]\!]\rho\}
\end{aligned}$$

Value Relation $\boxed{\mathcal{V}[\![\tau]\!]\rho}$

$$\begin{aligned}
\mathcal{V}[\![\tau_1 \xrightarrow{S} \tau_2]\!]\rho \triangleq \{\langle \lambda x.e, \sigma\rangle | \forall \rho', w \in \mathcal{W}[\![\tau_1]\!]\rho' : \rho'|_{\mathrm{FLoc}(\sigma)} = \rho|_{\mathrm{FLoc}(\sigma)}\\
\implies (e, (x, w) :: \sigma, S) \in \mathcal{X}[\![\tau_2]\!]\rho'\}
\end{aligned}$$

Weak Value Relation $\boxed{\mathcal{W}[\![\tau]\!]\rho}$

$$\mathcal{W}[\![\tau]\!]\rho \triangleq \mathcal{V}[\![\tau]\!]\rho \cup \{\mu\ell.v | v \in \mathcal{V}[\![\tau]\!]\rho[\ell \mapsto \tau]\}$$

Expression Relation $\boxed{\mathcal{X}[\![\tau]\!]\rho}$

$$\mathcal{X}[\![\tau]\!]\rho \triangleq \{(e, \sigma, S) | \mathrm{unsafe}(\sigma) \cap S = \varnothing \implies \mathrm{eval}(e, \sigma) \subseteq \mathcal{V}[\![\tau]\!]\rho\}$$

Semantic Typing $\boxed{\Gamma \vDash e : \tau, S}$

$$\Gamma \vDash e : \tau, S \triangleq \forall \rho, \sigma \in C[\![\Gamma]\!]\rho : (e, \sigma, S) \in \mathcal{X}[\![\tau]\!]\rho$$

We prove

$$\Gamma \vdash e : \tau, S \implies \Gamma \vDash e : \tau, S$$

by induction on $\vdash$.

Note that we have to extend the big-step evaluation rules with error propagation rules for $\mathrm{eval}(e, \sigma) \subseteq \mathcal{V}[\![\tau]\!]\rho$ to mean that $\mathrm{eval}(e, \sigma)$ has no errors. □

$$\begin{aligned}
\mathrm{unsafe}(\mathsf{Init}) &\triangleq \varnothing & \mathrm{unsafe}(\mathsf{Read}(E, x)) &\triangleq \mathrm{unsafe}(E)\\
\mathrm{unsafe}(\mathsf{Call}(E, v)) &\triangleq \mathrm{unsafe}(E) \cup \mathrm{unsafe}(v) & \mathrm{unsafe}([E]) &\triangleq \mathrm{unsafe}(E)\\
\mathrm{unsafe}(\bullet) &\triangleq \varnothing & \mathrm{unsafe}((x, \ell) :: \sigma) &\triangleq \{x\} \cup \mathrm{unsafe}(\sigma)\\
\mathrm{unsafe}((x, w) :: \sigma) &\triangleq \mathrm{unsafe}(w) \cup \mathrm{unsafe}(\sigma) & \mathrm{unsafe}(\langle \lambda x.e, \sigma\rangle) &\triangleq \mathrm{unsafe}(\sigma)\\
\mathrm{unsafe}(\mu\ell.v) &\triangleq \mathrm{unsafe}(v[\bullet/\ell])
\end{aligned}$$

$$\mathrm{eval}^\#(e, \Gamma) \triangleq \bigcup_S \{\tau | \Gamma \vdash e : \tau, S\} \quad \gamma(\tau) \triangleq \{v | v \in \mathcal{V}[\![\tau]\!]\bot \land \mathrm{unsafe}(v) = \varnothing\} \quad \gamma(T) \triangleq \bigcap_{\tau \in T} \gamma(\tau)$$

$$\mathrm{eval}(e, \gamma(\Gamma)) \subseteq \gamma(\mathrm{eval}^\#(e, \Gamma))$$

Test compilation