A Simple Abstract Interpretation Framework for Modular Analysis

JOONHYUP LEE and KWANGKEUN YI

1 SYNTAX AND SEMANTICS

1.1 Abstract Syntax

Identifiers	x	$\in$	Var	
Expression	e	$\longrightarrow$	$x \mid \lambda x.e \mid e\ e$	λ-calculus
		$\mid$	$e \rtimes e$	linked expression
		$\mid$	ε	empty module
		$\mid$	$x = e\ ;\ e$	(recursive) binding

Fig. 1. Abstract syntax of the language.

1.2 Operational Semantics

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

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:

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

Authors' address: Joonhyup Lee; Kwangkeun Yi.

XXXX-XXXX/2024/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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

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

$$\textsc{App}\ \frac{\sigma \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle \qquad \sigma \vdash e_2 \Downarrow v_2 \qquad (x, v_2) :: \sigma_1 \vdash e \Downarrow v}{\sigma \vdash e_1\ e_2 \Downarrow v}$$

$$\textsc{Link}\ \frac{\sigma \vdash e_1 \Downarrow \sigma_1 \qquad \sigma_1 \vdash e_2 \Downarrow v}{\sigma \vdash e_1 \rtimes e_2 \Downarrow v} \qquad \textsc{Empty}\ \frac{}{\sigma \vdash \varepsilon \Downarrow \bullet}$$

$$\textsc{Bind}\ \frac{\ell \notin \mathrm{FLoc}(\sigma) \qquad (x, \ell) :: \sigma \vdash e_1 \Downarrow v_1 \qquad (x, \mu\ell.v_1) :: \sigma \vdash e_1 \Downarrow \sigma_2}{\sigma \vdash x = e_1; e_2 \Downarrow (x, \mu\ell.v_1) :: \sigma_2}$$

Fig. 3. The big-step operational semantics.

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

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.

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

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

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

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

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

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

2 GENERATING AND RESOLVING EVENTS

Now we formulate the semantics for generating events.

Event	E	$\rightarrow$	Init	initial environment
		$\mid$	$\mathsf{Read}(E, x)$	read event
		$\mid$	$\mathsf{Call}(E, v)$	call event
Environment	σ	$\rightarrow$	$\cdots$	
		$\mid$	$[E]$	answer to an event
Value	v	$\rightarrow$	$\cdots$	
		$\mid$	E	answer to an event

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.

$$\bullet(x) \triangleq \bot \qquad ((x', \ell) :: \sigma)(x) \triangleq (x = x' ? \ell : \sigma(x))$$
$$[E](x) \triangleq \mathsf{Read}(E, x) \qquad ((x', w) :: \sigma)(x) \triangleq (x = x' ? w : \sigma(x))$$

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

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

Now we need to formulate the *concrete linking* rules. The concrete linking rule $\sigma_0 \rightarrowtail w$, 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:

$$\mathrm{eval}(e, \sigma) \triangleq \{v \mid \sigma \vdash e \Downarrow v\} \qquad \mathrm{eval}(e, \Sigma) \triangleq \bigcup_{\sigma \in \Sigma} \mathrm{eval}(e, \sigma) \qquad \sigma_0 \rightarrowtail W \triangleq \bigcup_{w \in W} (\sigma_0 \rightarrowtail w)$$

Then the following holds:

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

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

$$\boxed{\bowtie \in \mathsf{Env} \to \mathsf{Event} \to \mathscr{P}(\mathsf{Val})}$$

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

$$\boxed{\bowtie \in \mathsf{Env} \to \mathsf{Env} \to \mathscr{P}(\mathsf{Env})}$$

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

$$\boxed{\bowtie \in \mathsf{Env} \to \mathsf{Val} \to \mathscr{P}(\mathsf{Val})}$$

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

$$\boxed{\bowtie \in \mathsf{Env} \to \mathsf{WVal} \to \mathscr{P}(\mathsf{WVal})}$$

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

Fig. 7. Definition for concrete linking.

3 TYPING

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

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

Fig. 8. Definition of types.

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

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

$$\text{T-AppEvent}\ \frac{\Gamma \vdash e_1 : E^\#, S_1 \quad \Gamma \vdash e_2 : \tau, S_2}{\Gamma \vdash e_1\ e_2 : \text{Call}^\#(E^\#, \tau), S_1 \cup S_2} \qquad \text{T-Link}\ \frac{\Gamma \vdash e_1 : \Gamma_1, S_1 \quad \Gamma_1 \vdash e_2 : \tau_2, S_2}{\Gamma \vdash e_1 \bowtie e_2 : \tau_2, S_1 \cup S_2} \qquad \text{T-Nil}\ \frac{}{\Gamma \vdash \varepsilon : \bullet, \varnothing}$$

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

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

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

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

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

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

$$\mathcal{V}[\![\tau_1\xrightarrow{S}\tau_2]\!]\rho \triangleq \{\langle\lambda x.e,\sigma\rangle|\forall v\in\mathcal{V}[\![\tau_1]\!]\rho:(e,(x,v)::\sigma,S)\in\mathcal{X}[\![\tau_2]\!]\rho\}$$

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

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

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

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

by induction on $\vdash$.

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

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

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

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

REFERENCES