

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	\rightarrow	$x \mid \lambda x.e \mid e e$	λ -calculus
			$ e \bowtie e$	linked expression
			$ \varepsilon$	empty module
			$ 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	\rightarrow	\bullet	empty stack
			$ (x, \ell) :: \sigma$	free location binding
			$ (x, w) :: \sigma$	weak value binding
Value	v	\rightarrow	σ	exported environment
			$ \langle \lambda x.e, \sigma \rangle$	closure
Weak Value	w	\rightarrow	v	value
			$ \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:

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

Authors' address: Joonhyup Lee; Kwangkeun Yi.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Association for Computing Machinery.

XXXX-XXXX/2024/3-ART \$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

$$\begin{array}{c}
\boxed{\sigma \vdash e \Downarrow v} \\
\\
\text{ID} \quad \frac{\sigma(x) = v}{\sigma \vdash x \Downarrow v} \quad \text{RECI} \quad \frac{\sigma(x) = \mu\ell.v}{\sigma \vdash x \Downarrow v[\mu\ell.v/\ell]} \quad \text{FN} \quad \frac{}{\sigma \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle} \quad \text{APP} \quad \frac{\sigma \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle \quad \sigma \vdash e_2 \Downarrow v_2 \quad (x, v_2) :: \sigma_1 \vdash e \Downarrow v}{\sigma \vdash e_1 e_2 \Downarrow v} \\
\\
\text{LINK} \quad \frac{\sigma \vdash e_1 \Downarrow \sigma_1 \quad \sigma_1 \vdash e_2 \Downarrow v}{\sigma \vdash e_1 \bowtie e_2 \Downarrow v} \quad \text{EMPTY} \quad \frac{}{\sigma \vdash \varepsilon \Downarrow \bullet} \quad \text{BIND} \quad \frac{\ell \notin \text{FLoc}(\sigma) \quad (x, \ell) :: \sigma \vdash e_1 \Downarrow v_1 \quad (x, \mu\ell.v_1) :: \sigma \vdash e_2 \Downarrow \sigma_2}{\sigma \vdash x = e_1; e_2 \Downarrow (x, \mu\ell.v_1) :: \sigma_2}
\end{array}$$

Fig. 3. The big-step operational semantics.

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

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

and the following theorem:

THEOREM 1.1 (EQUIVALENCE OF SEMANTICS). For all $\sigma \in \text{Env}$, $\sigma' \in \text{MEnv} \times \text{Mem} \times \mathcal{P}(\text{Loc})$, $v \in \text{Val}$, $v' \in \text{MVal} \times \text{Mem} \times \mathcal{P}(\text{Loc})$, we have:

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

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

$$\begin{array}{c}
\boxed{\sigma, m, L \vdash e \Downarrow v, m', L'} \\
\\
\text{ID} \quad \frac{\sigma(x) = \ell \quad m(\ell) = v}{\sigma, m, L \vdash x \Downarrow v, m, L} \quad \text{FN} \quad \frac{}{\sigma, m, L \vdash \lambda x.e \Downarrow \langle \lambda x.e, \sigma \rangle, m, L} \\
\\
\text{APP} \quad \frac{\sigma, m, L \vdash e_1 \Downarrow \langle \lambda x.e, \sigma_1 \rangle, m_1, L_1 \quad \sigma, m_1, L_1 \vdash e_2 \Downarrow v_2, m_2, L_2 \quad \ell \notin \text{dom}(m_2) \cup L_2 \quad (x, \ell) :: \sigma_1, m_2[\ell \mapsto v_2], L_2 \vdash e \Downarrow v, m', L'}{\sigma, m, L \vdash e_1 e_2 \Downarrow v, m', L'} \\
\\
\text{LINK} \quad \frac{\sigma, m, L \vdash e_1 \Downarrow \sigma_1, m_1, L_1 \quad \sigma_1, m_1, L_1 \vdash e_2 \Downarrow v, m', L'}{\sigma, m, L \vdash e_1 \bowtie e_2 \Downarrow v, m', L'} \quad \text{EMPTY} \quad \frac{}{\sigma, m, L \vdash \varepsilon \Downarrow \bullet, m, L} \\
\\
\text{BIND} \quad \frac{\ell \notin \text{dom}(m) \cup L \quad (x, \ell) :: \sigma, m, L \cup \{\ell\} \vdash e_1 \Downarrow v_1, m_1, L_1 \quad (x, \ell) :: \sigma, m_1[\ell \mapsto v_1], L_1 \vdash e_2 \Downarrow \sigma_2, m', L'}{\sigma, m, L \vdash x = e_1; e_2 \Downarrow (x, \ell) :: \sigma_2, m', L'}
\end{array}$$

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
			Read(E, x)	read event
			Call(E, v)	call event
Environment	σ	\rightarrow	\dots	
			$[E]$	answer to an event
Value	v	\rightarrow	\dots	
			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.

$$\begin{aligned}
 \bullet(x) &\triangleq \perp & ((x', \ell) :: \sigma)(x) &\triangleq (x = x' ? \ell : \sigma(x)) \\
 [E](x) &\triangleq \text{Read}(E, x) & ((x', w) :: \sigma)(x) &\triangleq (x = x' ? w : \sigma(x))
 \end{aligned}$$

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

$$\frac{\text{APPEVENT} \quad \sigma \vdash e_1 \Downarrow E \quad \sigma \vdash e_2 \Downarrow v}{\sigma \vdash e_1 e_2 \Downarrow \text{Call}(E, v)}$$

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

$$\text{eval}(e, \sigma) \triangleq \{v \mid \sigma \vdash e \Downarrow v\} \quad \text{eval}(e, \Sigma) \triangleq \bigcup_{\sigma \in \Sigma} \text{eval}(e, \sigma) \quad \sigma_0 \bowtie W \triangleq \bigcup_{w \in W} (\sigma_0 \bowtie w)$$

Then the following holds:

THEOREM 2.1 (SOUNDNESS OF CONCRETE LINKING). Given $e \in \text{Expr}$, $\sigma \in \text{Env}$, $v \in \text{Val}$,

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

$$\bowtie \in \text{Env} \rightarrow \text{Event} \rightarrow \mathcal{P}(\text{Val})$$

$$\sigma_0 \bowtie \text{Init} \triangleq \{\sigma_0\}$$

$$\sigma_0 \bowtie \text{Read}(E, x) \triangleq \{v_+ | \sigma_+ \in \sigma_0 \bowtie E \wedge \sigma_+(x) = v_+\}$$

$$\cup \{v_+ [\mu\ell.v_+/\ell] | \sigma_+ \in \sigma_0 \bowtie E \wedge \sigma_+(x) = \mu\ell.v_+\}$$

$$\sigma_0 \bowtie \text{Call}(E, v) \triangleq \{v'_+ | \langle \lambda x.e, \sigma_+ \rangle \in \sigma_0 \bowtie E \wedge v_+ \in \sigma_0 \bowtie v \wedge (x, v_+) :: \sigma_+ \vdash e \Downarrow v'_+\}$$

$$\cup \{\text{Call}(E_+, v_+) | E_+ \in \sigma_0 \bowtie E \wedge v_+ \in \sigma_0 \bowtie v\}$$

$$\bowtie \in \text{Env} \rightarrow \text{Env} \rightarrow \mathcal{P}(\text{Env})$$

$$\sigma_0 \bowtie \bullet \triangleq \{\bullet\}$$

$$\sigma_0 \bowtie (x, \ell) :: \sigma \triangleq \{(x, \ell) :: \sigma_+ | \sigma_+ \in \sigma_0 \bowtie \sigma\}$$

$$\sigma_0 \bowtie (x, w) :: \sigma \triangleq \{(x, w_+) :: \sigma_+ | w_+ \in \sigma_0 \bowtie w \wedge \sigma_+ \in \sigma_0 \bowtie \sigma\}$$

$$\sigma_0 \bowtie [E] \triangleq \{\sigma_+ | \sigma_+ \in \sigma_0 \bowtie E\} \cup \{[E_+] | E_+ \in \sigma_0 \bowtie E\}$$

$$\bowtie \in \text{Env} \rightarrow \text{Val} \rightarrow \mathcal{P}(\text{Val})$$

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

$$\bowtie \in \text{Env} \rightarrow \text{WVal} \rightarrow \mathcal{P}(\text{WVal})$$

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

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

Fig. 8. Definition of types.

$\boxed{\Gamma \vdash e : \tau, S}$			
T-ID	T-FN	T-APP	
$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau, \{x\}}$	$\frac{(x, \tau_1) :: \Gamma \vdash e : \tau_2, S}{\Gamma \vdash \lambda x. e : \tau_1 \xrightarrow{S} \tau_2, \emptyset}$	$\frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{S} \tau_2, S_1 \quad \Gamma \vdash e_2 : \tau_1, S_2}{\Gamma \vdash e_1 e_2 : \tau_2, S_1 \cup S_2 \cup S}$	
T-APPEVENT	T-LINK	T-NIL	
$\frac{\Gamma \vdash e_1 : E^\#, S_1 \quad \Gamma \vdash e_2 : \tau, S_2}{\Gamma \vdash e_1 e_2 : \text{Call}^\#(E^\#, \tau), S_1 \cup S_2}$	$\frac{\Gamma \vdash e_1 : \Gamma_1, S_1 \quad \Gamma_1 \vdash e_2 : \tau_2, S_2}{\Gamma \vdash e_1 \bowtie e_2 : \tau_2, S_1 \cup S_2}$	$\frac{}{\Gamma \vdash \varepsilon : \bullet, \emptyset}$	
T-BIND			
$\frac{(x, \tau_1) :: \Gamma \vdash e_1 : \tau_1, S_1 \quad x \notin S_1 \quad (x, \tau_1) :: \Gamma \vdash e_2 : \tau_2, S_2}{\Gamma \vdash x = e_1; e_2 : (x, \tau_1) :: \tau_2, S_1 \cup S_2}$			

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

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

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

Context Relation

$$C[\![\Gamma]\!]\rho$$

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

Value Relation

$$\mathcal{V}[\![\tau]\!]\rho$$

$$\begin{aligned}\mathcal{V}[\![\tau_1 \xrightarrow{S} \tau_2]\!]\rho &\triangleq \{\langle \lambda x.e, \sigma \rangle \mid \forall \rho', w \in \mathcal{W}[\![\tau_1]\!]\rho' : \rho' \upharpoonright_{\text{FLoc}(\sigma)} = \rho \upharpoonright_{\text{FLoc}(\sigma)} \\ &\quad \Rightarrow (e, (x, w) :: \sigma, S) \in \mathcal{X}[\![\tau_2]\!]\rho'\}\end{aligned}$$

Weak Value Relation

$$\mathcal{W}[\![\tau]\!]\rho$$

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

Expression Relation

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

$$\mathcal{X}[\![\tau]\!]\rho \triangleq \{(e, \sigma, S) \mid \text{unsafe}(\sigma) \cap S = \emptyset \Rightarrow \text{eval}(e, \sigma) \subseteq \mathcal{V}[\![\tau]\!]\rho\}$$

Semantic Typing

$$\Gamma \models e : \tau, S$$

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

We prove

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

by induction on \vdash .

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

$$\text{unsafe}(\text{Init}) \triangleq \emptyset$$

$$\text{unsafe}(\text{Read}(E, x)) \triangleq \text{unsafe}(E)$$

$$\text{unsafe}(\text{Call}(E, v)) \triangleq \text{unsafe}(E) \cup \text{unsafe}(v)$$

$$\text{unsafe}([E]) \triangleq \text{unsafe}(E)$$

$$\text{unsafe}(\bullet) \triangleq \emptyset$$

$$\text{unsafe}((x, \ell) :: \sigma) \triangleq \{x\} \cup \text{unsafe}(\sigma)$$

$$\text{unsafe}((x, w) :: \sigma) \triangleq \text{unsafe}(w) \cup \text{unsafe}(\sigma)$$

$$\text{unsafe}(\langle \lambda x.e, \sigma \rangle) \triangleq \text{unsafe}(\sigma)$$

$$\text{unsafe}(\mu\ell.v) \triangleq \text{unsafe}(v[\bullet/\ell])$$

$$\text{eval}^\#(e, \Gamma) \triangleq \bigcup_S \{\tau \mid \Gamma \vdash e : \tau, S\} \quad \gamma(\tau) \triangleq \{v \mid v \in \mathcal{V}[\![\tau]\!]\perp \wedge \text{unsafe}(v) = \emptyset\} \quad \gamma(T) \triangleq \bigcap_{\tau \in T} \gamma(\tau)$$

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

REFERENCES