Soundness of Bottom-Up Type-State Analysis

Joonhyup Lee

February 14, 2024

1 Syntax and Semantics

1.1 Abstract Syntax

Identifiers	x, y	$\in$	Var	
Allocation site	h	$\in$	AllocSite	
Primitive methods	m	$\in$	$\mathbb{M}$	
Commands	C	$\to$	c	atomic commands
		$\mid$	$C + C$	branching
		$\mid$	$C; C$	sequencing
		$\mid$	C^*	iteration
Atomic commands	c	$\to$	$x = \text{new } h$	allocation
		$\mid$	$x = y$	assignment
		$\mid$	$x.m()$	primitive method call

Figure 1: Abstract syntax of the language.

1.2 Operational Semantics

Location	ℓ	$\in$	$\mathbb{L} \triangleq \text{AllocSite} \times \mathbb{N}$
Type-state	t	$\in$	$\mathbb{T} \triangleq \{\text{init, error, opened, closed}\}$
Interpretation of methods	$\underline{m}$	$\in$	$\mathbb{T} \to \mathbb{T}$
Environment	σ	$\in$	$\text{Env} \triangleq \text{Var} \xrightarrow{\text{fin}} \mathbb{L}$
Heap	H	$\in$	$\text{Heap} \triangleq \mathbb{L} \xrightarrow{\text{fin}} \mathbb{T}$

Figure 2: Definition of the semantic domains.

$$\boxed{\sigma, H \vdash C \Rightarrow \sigma, H}$$

$$\textsc{Br-L}\ \frac{\sigma, H \vdash C_1 \Rightarrow \sigma', H'}{\sigma, H \vdash C_1 + C_2 \Rightarrow \sigma', H'} \qquad \textsc{Br-R}\ \frac{\sigma, H \vdash C_2 \Rightarrow \sigma', H'}{\sigma, H \vdash C_1 + C_2 \Rightarrow \sigma', H'} \qquad \textsc{Seq}\ \frac{\sigma, H \vdash C_1 \Rightarrow \sigma_1, H_1 \qquad \sigma_1, H_1 \vdash C_2 \Rightarrow \sigma', H'}{\sigma, H \vdash C_1; C_2 \Rightarrow \sigma', H'}$$

$$\textsc{Iter-0}\ \frac{}{\sigma, H \vdash C^* \Rightarrow \sigma, H} \qquad \textsc{Iter-*}\ \frac{\sigma, H \vdash C^* \Rightarrow \sigma_1, H_1 \qquad \sigma_1, H_1 \vdash C \Rightarrow \sigma', H'}{\sigma, H \vdash C^* \Rightarrow \sigma', H'}$$

$$\textsc{Alloc}\ \frac{(h, n) = \ell \qquad \ell \notin \text{dom}(H)}{\sigma, H \vdash x = \text{new } h \Rightarrow \sigma[x \mapsto \ell], H[\ell \mapsto \text{init}]} \qquad \textsc{Assign}\ \frac{\sigma(y) = \ell}{\sigma, H \vdash x = y \Rightarrow \sigma[x \mapsto \ell], H} \qquad \textsc{Method}\ \frac{\sigma(x) = \ell \qquad H(\ell) = t}{\sigma, H \vdash v.m() \Rightarrow \sigma, H[\ell \mapsto \underline{m}(t)]}$$

Figure 3: Operational semantics of the language.

$$\textsc{Alloc-E}\ \frac{(h,n) = \ell \qquad \ell \notin \mathrm{dom}([H])}{[\sigma], [H], C \vdash x = \text{new } h \Rightarrow (x,\ell) :: [\sigma], (\ell, \text{init}) :: [H], C} \qquad \textsc{Assign-E}\ \frac{[\sigma](y) = \ell}{[\sigma], [H], C \vdash x = y \Rightarrow (x,\ell) :: [\sigma], [H], C}$$

$$\textsc{Method1-E}\ \frac{[\sigma](x) = [y] \qquad (t, C') \in [H]([y], C)}{[\sigma], [H], C \vdash v.m() \Rightarrow [\sigma], ([y], \underline{m}(t)) :: [H], C'} \qquad \textsc{Method2-E}\ \frac{[\sigma](x) = (h,n) \qquad [H](h,n) = t}{[\sigma], [H], C \vdash v.m() \Rightarrow [\sigma], ((h,n), \underline{m}(t)) :: [H], C}$$

Figure 4: Semantics for atomic commands, with read events.

Definition for reading from the heap under the set of constraints C:

$$\begin{aligned}
(([y], t) :: [H])([x], C) &\triangleq \{(t, \{[x] \doteq [y]\} \cup C)\} \cup [H]([y], \{[x] \dot{\neq} [y]\} \cup C) \\
(((h,n), _) :: [H])([x], C) &\triangleq [H]([x], C) \\
([])([x], C) &\triangleq \{(t, \{[[x]] \doteq t\} \cup C) | t \in \mathbb{T}\}
\end{aligned}$$