



모듈별 프로그램 따로 분석

이준협

2023년 8월 4일

ROPAS Show & Tell

Problem Statement

- Given two program segments e_1 and e_2 ,
- Want to derive a sound approximation of what $e_1 \bowtie e_2$ evaluates to, by *separately* analyzing e_1 and e_2 .

Overview

Abstract Syntax

Concrete Semantics

Concrete Linking

Abstract Syntax

Abstract Syntax

Identifiers	x, M	\in	Var	
Expression	e	\rightarrow	x	value identifier
			$\lambda x.e$	function
			$e e$	application
			$e \bowtie e$	linked expression
			ε	empty module
			M	module identifier
			$\text{let } x e e$	binding expression
			$\text{let } M e e$	binding module

Example

```
(* A.ml *)  
let true x y = x
```

```
(* main.ml *)  
open A  
let id x = x  
;;  
id true
```

```
(* In our language *)  
(let A =  
  let true = \x.\y.x in  $\varepsilon$   
in  $\varepsilon$ )  $\infty$   
  (A  $\infty$   
    ((let id = \x.x  
      in  $\varepsilon$ )  $\infty$   
      (id true)))
```

Concrete Semantics

Semantic Domains

Time	t	\in	\mathbb{T}
Context	C	\in	$\text{Ctx}(\mathbb{T})$
Value(Expr)	v	\in	$\text{Val}(\mathbb{T}) \triangleq \text{Expr} \times \text{Ctx}(\mathbb{T})$
Value(Expr/Mod)	V	\in	$\text{Val}(\mathbb{T}) \uplus \text{Ctx}(\mathbb{T})$
Memory	m	\in	$\text{Mem}(\mathbb{T}) \triangleq \mathbb{T} \xrightarrow{\text{fin}} \text{Val}(\mathbb{T})$
State	s	\in	$\text{State}(\mathbb{T}) \triangleq \text{Ctx}(\mathbb{T}) \times \text{Mem}(\mathbb{T}) \times \mathbb{T}$
Result	r	\in	$\text{Result}(\mathbb{T}) \triangleq (\text{Val}(\mathbb{T}) \uplus \text{Ctx}(\mathbb{T})) \times \text{Mem}(\mathbb{T}) \times \mathbb{T}$
Tick	tick	\in	$\text{Tick}(\mathbb{T}) \triangleq (\text{State}(\mathbb{T}) \times \text{Var} \times \text{Val}(\mathbb{T})) \rightarrow \mathbb{T}$
Context	C	\rightarrow	$[]$
		$ $	$(x, t) :: C$
		$ $	$(M, C) :: C$
Value(Expr)	v	\rightarrow	$\langle \lambda x. e, C \rangle$

addr and ctx

- $\text{addr}(C, x)$ looks up when x was bound in C .
- $\text{ctx}(C, M)$ looks up what is bound to M in C

$$\text{addr}(C, x) \triangleq \begin{cases} \perp & C = [] \\ t & C = (x, t) :: C' \\ \text{addr}(C', x) & C = (x', t) :: C' \wedge x' \neq x \\ \text{addr}(C'', x) & C = (M, C') :: C'' \end{cases}$$

$$\text{ctx}(C, M) \triangleq \begin{cases} \perp & C = [] \\ C' & C = (M, C') :: C'' \\ \text{ctx}(C'', M) & C = (M', C') :: C'' \wedge M' \neq M \\ \text{ctx}(C', M) & C = (x, t) :: C' \end{cases}$$

Operational Semantics : $\rightsquigarrow_{\text{tick}}$

$$(e, C, m, t) \rightsquigarrow_{\text{tick}} (V, m', t') \text{ or } (e', C', m', t')$$

$$[\text{EXPRVAR}] \frac{t_x = \text{addr}(C, x) \quad v = m(t_x)}{(x, C, m, t) \rightsquigarrow (v, m, t)}$$

$$[\text{APPL}] \frac{}{(e_1 e_2, C, m, t) \rightsquigarrow (e_1, C, m, t)} \quad [\text{APPR}] \frac{(e_1, C, m, t) \rightsquigarrow (\langle \lambda x. e_\lambda, C_\lambda \rangle, m_\lambda, t_\lambda)}{(e_1 e_2, C, m, t) \rightsquigarrow (e_2, C, m_\lambda, t_\lambda)}$$

$$[\text{APPBODY}] \frac{\begin{array}{c} (e_1, C, m, t) \rightsquigarrow (\langle \lambda x. e_\lambda, C_\lambda \rangle, m_\lambda, t_\lambda) \\ (e_2, C, m_\lambda, t_\lambda) \rightsquigarrow (v, m_a, t_a) \end{array}}{(e_1 e_2, C, m, t) \rightsquigarrow (e_\lambda, (x, t_a) : : C_\lambda, m_a[t_a \mapsto v], \text{tick}((C, m_a, t_a), x, v))}$$

T and tick

Q: So, what \mathbb{T} and tick should we use?

A: Anything! (If it satisfies two requirements)

\mathbb{T} and tick

Q: So, what \mathbb{T} and tick should we use?

A: Anything! (If it satisfies two requirements)

1. tick produces a strictly larger timestamp with respect to some total order on \mathbb{T} .

$$\forall t \in \mathbb{T} : t < \text{tick}((_, _, t), _, _)$$

2. All $(V, m, t) \in \text{Result}(\mathbb{T})$ satisfies: $V < t \wedge m < t$

$$C < t \triangleq \begin{cases} \text{True} & C = [] \\ t' < t \wedge C' < t & C = (x, t') :: C' \\ C' < t \wedge C'' < t & C = (M, C') :: C'' \end{cases} \quad V < t \triangleq \begin{cases} C < t & V = \langle _, C \rangle \\ C < t & V = C \end{cases}$$

$$m < t \triangleq \forall t' \in \text{dom}(m) : t' < t \wedge m(t') < t$$

Then all ticks will produce *fresh* timestamps.

Irrelevance of tick

Q: How can we consider semantics instantiated with different \mathbb{T} and tick to be equivalent?

A: If they have the same structure!

Irrelevance of tick

Q: How can we consider semantics instantiated with different \mathbb{T} and tick to be equivalent?

A: If they have the same structure!

- Let $r = (V, m, t) \in \text{Result}(\mathbb{T})$, $r' = (V', m', t') \in \text{Result}(\mathbb{T}')$.
- If we can *translate* the timestamps in r into r' and vice versa, r and r' are *isomorphic* ($r \cong r'$).

Irrelevance of tick

Q: How can we consider semantics instantiated with different \mathbb{T} and tick to be equivalent?

A: If they have the same structure!

- Let $r = (V, m, t) \in \text{Result}(\mathbb{T})$, $r' = (V', m', t') \in \text{Result}(\mathbb{T}')$.
- If we can *translate* the timestamps in r into r' and vice versa, r and r' are *isomorphic* ($r \cong r'$).
- “Can translate r to r' ”:
$$\exists f \in \mathbb{T} \rightarrow \mathbb{T}' : f(V) = V' \wedge f \circ m = m' \circ f \wedge f(t) = t'.$$

Irrelevance of tick

If we start from isomorphic states, the execution stays isomorphic!

Irrelevance of tick

If we start from isomorphic states, the execution stays isomorphic!

Notation: $\ell \in L \triangleq \text{Expr} \times \text{State}(\mathbb{T})$, $\rho \in R \triangleq L \cup \text{Result}(\mathbb{T})$

Theorem (Irrelevance of tick)

Let $s \in \text{State}(\mathbb{T})$ and $s' \in \text{State}(\mathbb{T}')$. If $s \cong s'$, then:

$$\forall \text{tick}, \text{tick}', e, \rho : (e, s) \rightsquigarrow_{\text{tick}} \rho \Rightarrow \exists \rho' : (e, s') \rightsquigarrow_{\text{tick}'} \rho' \wedge \rho \cong \rho'$$

Collecting Semantics

$$\llbracket e \rrbracket S \subseteq (R \times \text{Tick}(\mathbb{T})) \cup (L \times \text{Tick}(\mathbb{T}) \times R)$$

- Collecting semantics of e under $S \subseteq \text{State}(\mathbb{T}) \times \text{Tick}(\mathbb{T})$.
- Collect *all* configurations in the proof tree starting from $\{((e, s), \text{tick}) \mid (s, \text{tick}) \in S\} \subseteq R \times \text{Tick}(\mathbb{T})$.

Collecting Semantics

Definition (Transfer Function)

$$\text{Step}(A) \triangleq \left\{ \ell \rightsquigarrow_{\text{tick}} \rho, (\rho, \text{tick}) \mid \frac{A'}{\ell \rightsquigarrow_{\text{tick}} \rho} \wedge A' \subseteq A \wedge (\ell, \text{tick}) \in A \right\}$$

Definition (Collecting Semantics)

$$\llbracket e \rrbracket S \triangleq \text{lfp}(\lambda X. \text{Step}(X) \cup \{((e, s), \text{tick}) \mid (s, \text{tick}) \in S\})$$

Concrete Linking

Our Goal

What does $e_1 \bowtie e_2$ evaluate to under S ? : $\llbracket e_1 \bowtie e_2 \rrbracket S$

- Normally, first evaluate e_1 under S : $\llbracket e_1 \rrbracket S$
- Then evaluate e_2 under the exported states : $\llbracket e_2 \rrbracket \llbracket e_1 \rrbracket S$

$$\llbracket e_1 \bowtie e_2 \rrbracket S = \llbracket e_2 \rrbracket \llbracket e_1 \rrbracket S$$

- Instead, collect e_2 *in advance* under an *assumed* S_2 : $\llbracket e_2 \rrbracket S_2$
- Later, check if the assumption is guaranteed : $\llbracket e_1 \rrbracket S \cong S_1 \triangleright S_2$
- Fill in the blanks with S_1 , then finish : $S_1 \bowtie \llbracket e_2 \rrbracket S_2$

$$\llbracket e_1 \bowtie e_2 \rrbracket S \cong S_1 \bowtie \llbracket e_2 \rrbracket S_2$$

Definitions to Come

1. What is $|\llbracket e \rrbracket S|$ and $|S_1 \bowtie \llbracket e \rrbracket S_2|$ (final results)?

Definitions to Come

1. What is $\llbracket e \rrbracket S$ and $|S_1 \bowtie \llbracket e \rrbracket S_2|$ (final results)?

$$\llbracket e \rrbracket S \triangleq \{r \mid (e, _) \rightsquigarrow r \in \llbracket e \rrbracket S\} \quad |S_1 \bowtie \llbracket e \rrbracket S_2| \triangleq \{r \mid (e, _) \rightsquigarrow r \in S_1 \bowtie \llbracket e \rrbracket S_2\}$$

Definitions to Come

1. What is $\llbracket e \rrbracket S$ and $|S_1 \bowtie \llbracket e \rrbracket S_2|$ (final results)?

$$\llbracket e \rrbracket S \triangleq \{r \mid (e, _) \rightsquigarrow r \in \llbracket e \rrbracket S\} \quad |S_1 \bowtie \llbracket e \rrbracket S_2| \triangleq \{r \mid (e, _) \rightsquigarrow r \in S_1 \bowtie \llbracket e \rrbracket S_2\}$$

2. What is \triangleright (injection)?
3. What is \bowtie (semantic linking)?

Definition of \triangleright : Desired Properties

Elementwise injection: $(s_+, \text{tick}_+) = (s_1, \text{tick}_1) \triangleright (s_2, \text{tick}_2)$

1. If $s_2 = ([], \emptyset, 0)$, $s_+ \cong s_1$
2. If $(e, s_2) \rightsquigarrow_{\text{tick}_2}^* (e', s'_2)$, then $(s'_+, \text{tick}'_+) = (s_1, \text{tick}_1) \triangleright (s'_2, \text{tick}_2)$
must satisfy $\text{tick}_+ = \text{tick}'_+$ and $(e, s_+) \rightsquigarrow_{\text{tick}_+}^* (e', s'_+)$.

Definition of \triangleright , Step 1: Fill in the Blanks

$$C_2 \langle C_1 \rangle \triangleq \begin{cases} C_1 & C_2 = [] \\ (x, t) :: C' \langle C_1 \rangle & C_2 = (x, t) :: C' \\ (M, C' \langle C_1 \rangle) :: C'' \langle C_1 \rangle & C_2 = (M, C') :: C'' \end{cases}$$

$$C_2 \langle C_1 \rangle^{-1} \triangleq \begin{cases} [] & C_2 = C_1 \vee C_2 = [] \\ (x, t) :: C' \langle C_1 \rangle^{-1} & C_2 = (x, t) :: C' \\ (M, C' \langle C_1 \rangle^{-1}) :: C'' \langle C_1 \rangle^{-1} & C_2 = (M, C') :: C'' \end{cases}$$

Definition of \triangleright , Step 1: Fill in the Blanks

$$C_2 \langle C_1 \rangle \triangleq \begin{cases} C_1 & C_2 = [] \\ (x, t) :: C' \langle C_1 \rangle & C_2 = (x, t) :: C' \\ (M, C' \langle C_1 \rangle) :: C'' \langle C_1 \rangle & C_2 = (M, C') :: C'' \end{cases}$$

$$C_2 \langle C_1 \rangle^{-1} \triangleq \begin{cases} [] & C_2 = C_1 \vee C_2 = [] \\ (x, t) :: C' \langle C_1 \rangle^{-1} & C_2 = (x, t) :: C' \\ (M, C' \langle C_1 \rangle^{-1}) :: C'' \langle C_1 \rangle^{-1} & C_2 = (M, C') :: C'' \end{cases}$$

$$V_2 \langle C_1 \rangle \triangleq \begin{cases} C_2 \langle C_1 \rangle & V_2 = C_2 \\ \langle \lambda x. e, C_2 \langle C_1 \rangle \rangle & V_2 = \langle \lambda x. e, C_2 \rangle \end{cases} \quad m_2 \langle C_1 \rangle \triangleq \bigcup_{t \in \text{dom}(m_2)} \{t \mapsto m_2(t) \langle C_1 \rangle\}$$

$$r_2 \langle s_1 \rangle \triangleq (V_2 \langle C_1 \rangle, m_1 \cup m_2 \langle C_1 \rangle, t_2)$$

Definition of \triangleright , Step 2: tick_+

Note $C_2 \langle C_1 \rangle \in \text{Ctx}(\mathbb{T}_1 + \mathbb{T}_2)$. What should tick_+ be?

$$\text{tick}_+((C, m, t), x, v) \triangleq \begin{cases} \text{tick}_1((C.1, m.1, t), x, v.1) & t \in \mathbb{T}_1 \\ \text{tick}_2((C \langle C_1 \rangle^{-1}.2, m \langle C_1 \rangle^{-1}.2, t), x, v \langle C_1 \rangle^{-1}.2) & t \in \mathbb{T}_2 \end{cases}$$

where

$$C.i \triangleq \begin{cases} [] & C = [] \\ (x, t) :: C'.i & C = (x, t) :: C' \wedge t \in \mathbb{T}_i \\ C'.i & C = (x, t) :: C' \wedge t \notin \mathbb{T}_i \\ (M, C'.i) :: C''.i & C = (M, C') :: C'' \end{cases}$$

$$V.i \triangleq \begin{cases} C.i & V = C \\ \langle \lambda x.e, C.i \rangle & V = \langle \lambda x.e, C \rangle \end{cases}$$

$$m.i \triangleq \bigcup_{t \in \text{dom}(m) \cap \mathbb{T}_i} \{t \mapsto m(t).i\}$$

Definition of \triangleright

Definition (Definition of \triangleright)

$$(s_1, \text{tick}_1) \triangleright (r_2, \text{tick}_2) \triangleq (r_2 \langle s_1 \rangle, \text{tick}_+)$$

$$S_1 \triangleright S_2 \triangleq \{(s_1, \text{tick}_1) \triangleright (s_2, \text{tick}_2) \mid (s_1, \text{tick}_1) \in S_1 \wedge (s_2, \text{tick}_2) \in S_2\}$$

To inject into a cache,

$$(s_1, \text{tick}_1) \triangleright (\rho_2, \text{tick}_2) \triangleq \begin{cases} (r_+, \text{tick}_+) & \rho_2 = r_2 \wedge (r_+, \text{tick}_+) = (s_1, \text{tick}_1) \triangleright (r_2, \text{tick}_2) \\ ((e, s_+), \text{tick}_+) & \rho_2 = \ell_2 = (e, s_2) \wedge (s_+, \text{tick}_+) = (s_1, \text{tick}_1) \triangleright (s_2, \text{tick}_2) \end{cases}$$

$$(s_1, \text{tick}_1) \triangleright (\ell_2 \rightsquigarrow_{\text{tick}_2} \rho_2) \triangleq \ell_+ \rightsquigarrow_{\text{tick}_+} \rho_+$$

$$\text{where } (\ell_+, \text{tick}_+) = (s_1, \text{tick}_1) \triangleright (\ell_2, \text{tick}_2) \wedge (\rho_+, \text{tick}_+) = (s_1, \text{tick}_1) \triangleright (\rho_2, \text{tick}_2)$$

Definition of \bowtie

Definition (Semantic Linking)

Let $S_1 \subseteq \text{State}(\mathbb{T}_1) \times \text{Tick}(\mathbb{T}_1)$ and

$A_2 \subseteq (L_2 \times \text{Tick}(\mathbb{T}_2) \times R_2) \cup (R_2 \times \text{Tick}(\mathbb{T}_2))$. Then:

$$S_1 \bowtie A_2 \triangleq \text{lfp}(\lambda X. \text{Step}(X) \cup (S_1 \triangleright A_2))$$

Advance

Lemma (Advance)

Let $S_1 \subseteq \text{State}(\mathbb{T}_1) \times \text{Tick}(\mathbb{T}_1)$ and $S_2 \subseteq \text{State}(\mathbb{T}_2) \times \text{Tick}(\mathbb{T}_2)$.

Then:

$$\llbracket e \rrbracket (S_1 \triangleright S_2) = S_1 \bowtie \llbracket e \rrbracket S_2$$