A Simple Abstract Interpretation Framework for Modular Analysis

Joonhyup Lee, Kwangkeun Yi

Nov. 16, 2023

ROPAS@SNU

Problem Formulation

How do we formalize analyzing program fragments (exporting modules & the main expression) *in advance*?

```
(* Module M *)
let x = 1
```

```
(* Module F *)
let fix fact n =
  if n < 1 then 1
  else n * fact (n − 1)
```

```
(* Client code *)
Include M
Include F
(F.fact 100) + M.x
```

- Analyze with only `F` assumed ($100! + ?$)
- Then link `M` afterwards.

Problem Formulation

How do we formalize analyzing program fragments (exporting modules & the main expression) *in advance*?

```
(* Module M *)
let x = 1
```

```
(* Module F *)
let fix fact n =
  if n < 1 then 1
  else n * fact (n - 1)
```

```
(* Client code *)
Include M
Include F
(F.fact 100) + M.x
```

- Analyze with only `F` assumed ($100! + ?$)
- Then link `M` afterwards.

How to design separate, modular analysis, then link?

Table of Contents

A model language : Call-by-Value λ Calculus with Modules

Syntax

$$
\begin{array}{rrcll}
\text{Identifiers} & x, d & \in & \mathsf{Var} & \\
\text{Expression} & e & \rightarrow & x \mid \lambda x.e \mid e\ e & \text{untyped } \lambda\text{-calculus} \\
 & & \mid & e \rtimes e & \text{linked expression} \\
 & & \mid & \varepsilon & \text{empty module} \\
 & & \mid & d & \text{module identifier} \\
 & & \mid & \texttt{let}\ x\ e\ e & \text{expression binding} \\
 & & \mid & \texttt{let}\ d\ e\ e & \text{module binding}
\end{array}
$$

Semantic Domains

$$
\begin{array}{rcll}
\text{Environment/Context} & \sigma & \in & \mathsf{Ctx} \\
\text{Value of expressions} & v & \in & \mathsf{Val} \triangleq \mathsf{Var} \times \mathsf{Expr} \times \mathsf{Ctx} \\
\text{Value of expressions/modules} & V & \in & \mathsf{Val} + \mathsf{Ctx} \\
\text{Configuration (left)} & c & \in & \mathsf{Config} \triangleq \mathsf{Expr} \times \mathsf{Ctx} \\
\text{Configuration (right)} & r & \in & \mathsf{Right} \triangleq \mathsf{Config} + \mathsf{Val} + \mathsf{Ctx} \\
\text{Context} & \sigma & \to & [] \\
 & & | & (x, v) :: \sigma \\
 & & | & (d, \sigma) :: \sigma \\
\text{Value of expressions} & v & \to & \langle \lambda x.e, \sigma \rangle
\end{array}
$$

Operational Semantics (call-by-value λ calculus)

$$\boxed{(e, \sigma) \hookrightarrow V \text{ or } (e', \sigma')}$$

$$[\textsc{ExprID}]\ \frac{v = \sigma(x)}{(x, \sigma) \hookrightarrow v} \qquad [\textsc{Fn}]\ \frac{}{(\lambda x.e, \sigma) \hookrightarrow \langle \lambda x.e, \sigma \rangle} \qquad [\textsc{AppL}]\ \frac{}{(e_1\ e_2, \sigma) \hookrightarrow (e_1, \sigma)}$$

$$[\textsc{AppR}]\ \frac{(e_1, \sigma) \hookrightarrow \langle \lambda x.e_\lambda, \sigma_\lambda \rangle}{(e_1\ e_2, \sigma) \hookrightarrow (e_2, \sigma)} \qquad [\textsc{AppBody}]\ \frac{\begin{array}{c}(e_1, \sigma) \hookrightarrow \langle \lambda x.e_\lambda, \sigma_\lambda \rangle \\ (e_2, \sigma) \hookrightarrow v\end{array}}{(e_1\ e_2, \sigma) \hookrightarrow (e_\lambda, (x, v) :: \sigma_\lambda)}$$

$$[\textsc{App}]\ \frac{\begin{array}{c}(e_1, \sigma) \hookrightarrow \langle \lambda x.e_\lambda, \sigma_\lambda \rangle \\ (e_2, \sigma) \hookrightarrow v \\ (e_\lambda, (x, v) :: \sigma_\lambda) \hookrightarrow v'\end{array}}{(e_1\ e_2, \sigma) \hookrightarrow v'}$$

Operational Semantics (modules)

$$\boxed{(e, \sigma) \hookrightarrow V \text{ or } (e', \sigma')}$$

$$[\textsc{Empty}]\ \frac{}{(\varepsilon, \sigma) \hookrightarrow \sigma} \qquad [\textsc{ModID}]\ \frac{\sigma' = \sigma(d)}{(d, \sigma) \hookrightarrow \sigma'}$$

$$[\textsc{LetEL}]\ \frac{}{(\texttt{let}\ x\ e_1\ e_2, \sigma) \hookrightarrow (e_1, \sigma)} \quad [\textsc{LetER}]\ \frac{(e_1, \sigma) \hookrightarrow v}{(\texttt{let}\ x\ e_1\ e_2, \sigma) \hookrightarrow (e_2, (x, v) :: \sigma)}$$

$$[\textsc{LetML}]\ \frac{}{(\texttt{let}\ d\ e_1\ e_2, \sigma) \hookrightarrow (e_1, \sigma)} \quad [\textsc{LetMR}]\ \frac{(e_1, \sigma) \hookrightarrow \sigma'}{(\texttt{let}\ d\ e_1\ e_2, \sigma) \hookrightarrow (e_2, (d, \sigma') :: \sigma)}$$

$$[\textsc{LetE}]\ \frac{(e_1, \sigma) \hookrightarrow v \quad (e_2, (x, v) :: \sigma) \hookrightarrow \sigma'}{(\texttt{let}\ x\ e_1\ e_2, \sigma) \hookrightarrow \sigma'} \quad [\textsc{LetM}]\ \frac{(e_1, \sigma) \hookrightarrow \sigma' \quad (e_2, (d, \sigma') :: \sigma) \hookrightarrow \sigma''}{(\texttt{let}\ d\ e_1\ e_2, \sigma) \hookrightarrow \sigma''}$$

Operational Semantics (linking)

$$\boxed{(e,\sigma) \hookrightarrow V \text{ or } (e',\sigma')}$$

$$[\textsc{LinkL}]\ \frac{}{(e_1 \rtimes e_2, \sigma) \hookrightarrow (e_1, \sigma)} \quad [\textsc{LinkR}]\ \frac{(e_1,\sigma) \hookrightarrow \sigma'}{(e_1 \rtimes e_2, \sigma) \hookrightarrow (e_2, \sigma')} \quad [\textsc{Link}]\ \frac{(e_1,\sigma) \hookrightarrow \sigma' \quad (e_2,\sigma') \hookrightarrow V}{(e_1 \rtimes e_2, \sigma) \hookrightarrow V}$$

Collecting Semantics & Modularity

Collecting Semantics

$$\Sigma \triangleq \mathsf{Right} + \hookrightarrow \qquad \mathsf{Trace} \triangleq \mathscr{P}(\Sigma)$$

Definition (Transfer function)

Given $A \subseteq \Sigma$, define:

$$\mathsf{Step}(A) \triangleq \left\{ c \hookrightarrow r, r \middle| \frac{A'}{c \hookrightarrow r} \text{ and } A' \subseteq A \text{ and } c \in A \right\}$$

Definition (Collecting semantics)

Given $e \in \mathsf{Expr}$ and $C \subseteq \mathsf{Ctx}$, define:

$$[\![e]\!]C \triangleq \mathsf{lfp}(\lambda X.\mathsf{Step}(X) \cup \{(e, \sigma) | \sigma \in C\})$$

Modularity

Theorem (Modularity)

For all $e_1, e_2 \in \mathsf{Expr}$ and $C, C_1, C_2 \subseteq \mathsf{Ctx}$, we have:

$$[\![e_1 \rtimes e_2]\!]C = C_1 \infty [\![e_2]\!]C_2$$

where $C_1 \rhd C_2 = [\![e_1]\!]C$.

$[\![e_1]\!]C$ (export) and $[\![e_2]\!]C_2$ (client) are separate

The Modularity Theorem Hinges on the Advance Lemma

Lemma (Advance)

For all $e \in \mathsf{Expr}$ and $C_1, C_2 \subseteq \mathsf{Ctx}$, $[\![e]\!](C_1 \rhd C_2) = C_1 \infty [\![e]\!] C_2$.

Where:

Definition (Injection)

For $C \subseteq \mathsf{Ctx}$ and $A \subseteq \Sigma$, define:

$$C \rhd A \triangleq \{r\langle\sigma\rangle | \sigma \in C, r \in A\} \cup \{c\langle\sigma\rangle \hookrightarrow r\langle\sigma\rangle | \sigma \in C, c \hookrightarrow r \in A\}$$

Definition (Semantic Linking)

For $C \subseteq \mathsf{Ctx}$ and $A \subseteq \Sigma$, define:

$$C \infty A \triangleq \mathsf{lfp}(\lambda X.\mathsf{Step}(X) \cup (C \rhd A))$$

Injection

Lemma (Injection Preserves $\hookrightarrow$)

$\forall c \in \text{Config}, r \in \text{Right}, \sigma \in \text{Ctx}, c \hookrightarrow r \Rightarrow c\langle\sigma\rangle \hookrightarrow r\langle\sigma\rangle$

Where:

$$r_2\langle\sigma_1\rangle \triangleq \begin{cases} \sigma_1 & r_2 = [] \\ (x, v\langle\sigma_1\rangle) :: \sigma\langle\sigma_1\rangle & r_2 = (x, v) :: \sigma \\ (d, \sigma\langle\sigma_1\rangle) :: \sigma'\langle\sigma_1\rangle & r_2 = (d, \sigma) :: \sigma' \\ \langle\lambda x.e, \sigma\langle\sigma_1\rangle\rangle & r_2 = \langle\lambda x.e, \sigma\rangle \\ (e, \sigma\langle\sigma_1\rangle) & r_2 = (e, \sigma) \end{cases}$$

A Trivial Case

For the trivial case when $C_2 = \{[]\}$, we have:

Corollary

For all $e_1, e_2 \in \mathsf{Expr}$ and $C \subseteq \mathsf{Ctx}$, we have:

$$[\![e_1 \rtimes e_2]\!]C = ([\![e_1]\!]C) \propto [\![e_2]\!]\mathsf{emp}$$

where $\mathsf{emp} = \{[]\}$.

Skeleton for Static Analysis

Require $\mathsf{Trace}^{\#}, \mathsf{Step}^{\#}, \rhd^{\#}$:

$$\mathsf{Trace} = \mathscr{P}(\Sigma) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathsf{Trace}^{\#}$$

$$\mathsf{Step} \circ \gamma \subseteq \gamma \circ \mathsf{Step}^{\#} \qquad \rhd \circ (\gamma, \gamma) \subseteq \gamma \circ \rhd^{\#}$$

Define:

$$[\![e]\!]^{\#} C^{\#} \triangleq \mathsf{lfp}(\lambda X^{\#}.\mathsf{Step}^{\#}(X^{\#}) \cup^{\#} \alpha\{(e,\sigma) | \sigma \in \gamma C^{\#}\})$$

$$C^{\#} \propto^{\#} A^{\#} \triangleq \mathsf{lfp}(\lambda X^{\#}.\mathsf{Step}^{\#}(X^{\#}) \cup^{\#} (C^{\#} \rhd^{\#} A^{\#}))$$

So that:

$$[\![e]\!] \circ \gamma \subseteq \gamma \circ [\![e]\!]^{\#} \qquad \propto \circ (\gamma, \gamma) \subseteq \gamma \circ \propto^{\#}$$

Instrumented Collecting Semantics & Modularity

Semantic Domains

$$
\begin{array}{rccl}
\text{Time} & t & \in & \mathbb{T} \\
\text{Environment/Context} & \sigma & \in & \mathsf{Ctx} \\
\text{Value of expressions} & v & \in & \mathsf{Val} \triangleq \mathsf{Var} \times \mathsf{Expr} \times \mathsf{Ctx} \\
\text{Value of expressions/modules} & V & \in & \mathsf{Val} + \mathsf{Ctx} \\
\text{Memory} & m & \in & \mathsf{Mem} \triangleq \mathbb{T} \xrightarrow{\mathsf{fin}} \mathsf{Val} \\
\text{State} & s & \in & \mathsf{State} \triangleq \mathsf{Ctx} \times \mathsf{Mem} \times \mathbb{T} \\
\text{Outcome} & o & \in & \mathsf{Outcome} \triangleq (\mathsf{Val} + \mathsf{Ctx}) \times \mathsf{Mem} \times \mathbb{T} \\
\text{Configuration (left)} & c & \in & \mathsf{Config} \triangleq \mathsf{Expr} \times \mathsf{State} \\
\text{Configuration (right)} & r & \in & \mathsf{Right} \triangleq \mathsf{Config} + \mathsf{Outcome} \\
\text{Context} & \sigma & \to & [] \\
 & & \mid & (x, t) :: \sigma \\
 & & \mid & (d, \sigma) :: \sigma \\
\text{Value of expressions} & v & \to & \langle \lambda x.e, \sigma \rangle
\end{array}
$$

$\mathbb{T}$ **and** tick

Parametrized by choice of the set $\mathbb{T}$(Time) and the function tick.

- $\mathbb{T}$: Timestamps for program states, used as addresses.
- tick: Produces fresh timestamps.

Freshness: *total order* on $\mathbb{T}$

$$t < \mathsf{tick}(t)$$

$$\mathsf{State} \triangleq \{(\sigma, m, t) | \sigma \leq t \text{ and } m \leq t\}$$

$$\mathsf{Outcome} \triangleq \{(V, m, t) | V \leq t \text{ and } m \leq t\}$$

Operational Semantics (1/2)

$$\boxed{(e, \sigma, m, t) \hookrightarrow (V, m', t') \text{ or } (e', \sigma', m', t')}$$

$$[\textsc{ExprID}]\ \frac{t_x = \sigma(x) \qquad v = m(t_x)}{(x, \sigma, m, t) \hookrightarrow (v, m, t)} \qquad [\textsc{Fn}]\ \frac{}{(\lambda x.e, \sigma, m, t) \hookrightarrow (\langle \lambda x.e, \sigma \rangle, m, t)}$$

$$[\textsc{App}]\ \frac{\begin{array}{c}(e_1, \sigma, m, t) \hookrightarrow (\langle \lambda x.e_\lambda, \sigma_\lambda \rangle, m_\lambda, t_\lambda) \\ (e_2, \sigma, m_\lambda, t_\lambda) \hookrightarrow (v, m_a, t_a) \\ (e_\lambda, (x, \mathsf{tick}(t_a)) :: \sigma_\lambda, m_a[\mathsf{tick}(t_a) \mapsto v], \mathsf{tick}(t_a)) \hookrightarrow (v', m', t')\end{array}}{(e_1\ e_2, \sigma, m, t) \hookrightarrow (v', m', t')}$$

$$[\textsc{Link}]\ \frac{\begin{array}{c}(e_1, \sigma, m, t) \hookrightarrow (\sigma', m', t') \\ (e_2, \sigma', m', t') \hookrightarrow (V, m'', t'')\end{array}}{(e_1 \rtimes e_2, \sigma, m, t) \hookrightarrow (V, m'', t'')} \qquad [\textsc{Empty}]\ \frac{}{(\varepsilon, \sigma, m, t) \hookrightarrow (\sigma, m, t)}$$

Operational Semantics (2/2)

$$\boxed{(e, \sigma, m, t) \hookrightarrow (V, m', t') \text{ or } (e', \sigma', m', t')}$$

$$[\textsc{ModID}]\ \frac{\sigma' = \sigma(d)}{(d, \sigma, m, t) \hookrightarrow (\sigma', m, t)}$$

$$[\textsc{LetE}]\ \frac{\begin{array}{c}(e_1, \sigma, m, t) \hookrightarrow (v, m', t') \\ (e_2, (x, \mathsf{tick}(t')) :: \sigma, m'[\mathsf{tick}(t') \mapsto v], \mathsf{tick}(t')) \hookrightarrow (\sigma', m'', t'')\end{array}}{(\texttt{let}\ x\ e_1\ e_2, \sigma, m, t) \hookrightarrow (\sigma', m'', t'')}$$

$$[\textsc{LetM}]\ \frac{\begin{array}{c}(e_1, \sigma, m, t) \hookrightarrow (\sigma', m', t') \\ (e_2, (d, \sigma') :: \sigma, m', t') \hookrightarrow (\sigma'', m'', t'')\end{array}}{(\texttt{let}\ d\ e_1\ e_2, \sigma, m, t) \hookrightarrow (\sigma'', m'', t'')}$$

Collecting Semantics

$$\Sigma \triangleq \text{Right} + \hookrightarrow \qquad \text{Trace} \triangleq \mathscr{P}(\Sigma)$$

Definition (Transfer function)

Given $A \subseteq \Sigma$, define

$$\text{Step}(A) \triangleq \left\{ c \hookrightarrow r, r \middle| \frac{A'}{c \hookrightarrow r} \text{ and } A' \subseteq A \text{ and } c \in A \right\}$$

Definition (Collecting semantics)

Given $e \in \text{Expr}$ and $S \subseteq \text{State}$, define:

$$[\![e]\!]S \triangleq \text{lfp}(\lambda X.\text{Step}(X) \cup \{(e, s) | s \in S\})$$

Linking Timestamps

$$\mathbb{T}_\infty \triangleq \mathbb{T}_1 + \mathbb{T}_2 \quad \leq_\infty \triangleq \text{lexicographic order} \quad \text{tick}_\infty(t) \triangleq \begin{cases} \text{tick}_1(t) & t \in \mathbb{T}_1 \\ \text{tick}_2(t) & t \in \mathbb{T}_2 \end{cases}$$

Notation

All sets with the subscript $i (i = 1, 2)$ is assumed to be using $\mathbb{T}_i$ as timestamps, and all sets with the subscript ∞ is assumed to be using $\mathbb{T}_\infty$ as timestamps.

Injection

Lemma (Injection Preserves $\hookrightarrow$)

For all $s_1 \in \mathsf{State}_1$, $c_2 \in \mathsf{Config}_2$, $r_2 \in \mathsf{Right}_2$,

$$c_2 \hookrightarrow_2 r_2 \Rightarrow c_2\langle s_1 \rangle \hookrightarrow_\infty r_2\langle s_1 \rangle$$

Where:

$$V_2\langle \sigma_1 \rangle \triangleq \begin{cases} \sigma_1 & V_2 = [] \\ (x,t) :: \sigma\langle \sigma_1 \rangle & V_2 = (x,t) :: \sigma \\ (d, \sigma\langle \sigma_1 \rangle) :: \sigma'\langle \sigma_1 \rangle & V_2 = (d, \sigma) :: \sigma' \\ \langle \lambda x.e, \sigma_2\langle \sigma_1 \rangle \rangle & V_2 = \langle \lambda x.e, \sigma_2 \rangle \end{cases}$$

$$m_2\langle \sigma_1 \rangle \triangleq \bigcup_{t \in \mathsf{dom}(m_2)} \{t \mapsto m_2(t)\langle \sigma_1 \rangle\}$$

$$o_2\langle s_1 \rangle \triangleq (V_2\langle \sigma_1 \rangle, m_1 \cup m_2\langle \sigma_1 \rangle, t_2)$$

Semantic Linking

Definition (Injection)

For $S_1 \subseteq \mathsf{State}_1$ and $A_2 \subseteq \Sigma_2$, define:

$$\begin{aligned} S_1 \rhd A_2 \triangleq & \{r_2\langle s_1\rangle | s_1 \in S_1, r_2 \in A_2\} \cup \\ & \{c_2\langle s_1\rangle \hookrightarrow_{\infty} r_2\langle s_1\rangle | s_1 \in S_1, c_2 \hookrightarrow_2 r_2 \in A_2\} \end{aligned}$$

Definition (Semantic Linking)

For $S_1 \subseteq \mathsf{State}_1$ and $A_2 \subseteq \Sigma_2$, define:

$$S_1 \infty A_2 \triangleq \mathsf{lfp}(\lambda X.\mathsf{Step}_{\infty}(X) \cup (S_1 \rhd A_2))$$

Advance Lemma

Lemma (Advance)

For all $e \in \mathsf{Expr}$ and $S_1 \subseteq \mathsf{State}_1$, $S_2 \subseteq \mathsf{State}_2$,

$$[\![e]\!](S_1 \rhd S_2) = S_1 \propto [\![e]\!] S_2$$

The Same Modularity Theorem

Theorem (Modularity)

For all $e_1, e_2 \in \mathsf{Expr}$ and $S \subseteq \mathsf{State}, S_i \subseteq \mathsf{State}_i (i = 1, 2)$, we have:

$$[\![e_1 \rtimes e_2]\!]S \cong S_1 \propto [\![e_2]\!]S_2$$

where $S_1 \triangleright S_2 \cong [\![e_1]\!]S$.

Note, no longer: $[\![e_1 \rtimes e_2]\!]S = ([\![e_1]\!]S) \propto [\![e_2]\!]\mathsf{emp}$

since $\underbrace{[\![e_1]\!]S \triangleright \mathsf{emp}}_{\text{linked timestamp}} \neq \underbrace{[\![e_1]\!]S}_{\text{not linked}}$.

$\boxed{\text{Need to replace } = \text{ with } \cong}$

$\boxed{\cong\text{: Defined later}}$

Abstracting the Instrumented Collecting Semantics & Modular Analysis

Semantic Domains

$$
\begin{array}{rccl}
\text{Abstract Time} & \dot{t} & \in & \dot{\mathbb{T}} \\
\text{Environment/Context} & \dot{\sigma} & \in & \dot{\mathsf{Ctx}} \\
\text{Value of expressions} & \dot{v} & \in & \dot{\mathsf{Val}} \triangleq \mathsf{Var} \times \mathsf{Expr} \times \dot{\mathsf{Ctx}} \\
\text{Value of expressions/modules} & \dot{V} & \in & \dot{\mathsf{Val}} + \dot{\mathsf{Ctx}} \\
\text{Abstract Memory} & \dot{m} & \in & \dot{\mathsf{Mem}} \triangleq \dot{\mathbb{T}} \xrightarrow{\text{fin}} \mathscr{P}(\dot{\mathsf{Val}}) \\
\text{Abstract State} & \dot{s} & \in & \dot{\mathsf{State}} \triangleq \dot{\mathsf{Ctx}} \times \dot{\mathsf{Mem}} \times \dot{\mathbb{T}} \\
\text{Abstract outcome} & \dot{o} & \in & \dot{\mathsf{Outcome}} \triangleq (\dot{\mathsf{Val}} + \dot{\mathsf{Ctx}}) \times \dot{\mathsf{Mem}} \times \dot{\mathbb{T}} \\
\text{Abstract configuration (left)} & \dot{c} & \in & \dot{\mathsf{Config}} \triangleq \mathsf{Expr} \times \dot{\mathsf{State}} \\
\text{Abstract configuration (right)} & \dot{r} & \in & \dot{\mathsf{Right}} \triangleq \dot{\mathsf{Config}} + \dot{\mathsf{Outcome}} \\
\text{Context} & \dot{\sigma} & \rightarrow & [] \\
 & & \mid & (x, \dot{t}) :: \dot{\sigma} \\
 & & \mid & (d, \dot{\sigma}) :: \dot{\sigma} \\
\text{Value of expressions} & \dot{v} & \rightarrow & \langle \lambda x.e, \dot{\sigma} \rangle
\end{array}
$$

$\mathbb{T}$ **and** tick

Parametrized by choice of the set $\dot{\mathbb{T}}$(Time) and the function $\dot{\mathsf{tick}}$.

- $\dot{\mathbb{T}}$: Timestamps for program states, used as addresses.
- $\dot{\mathsf{tick}}$: Satisfies $\dot{\alpha} \circ \mathsf{tick} = \dot{\mathsf{tick}} \circ \dot{\alpha}$.

Operational Semantics (that differ from the concrete)

$$\boxed{(e, \dot{\sigma}, \dot{m}, \dot{t}\,) \mathrel{\dot{\hookrightarrow}} (\dot{V}, \dot{m}', \dot{t}') \text{ or } (e', \dot{\sigma}', \dot{m}', \dot{t}')}$$

$$[\textsc{ExprID}] \quad \frac{\dot{t}_x = \dot{\sigma}(x) \qquad \dot{v} \in \dot{m}(\dot{t}_x)}{(x, \dot{\sigma}, \dot{m}, \dot{t}\,) \mathrel{\dot{\hookrightarrow}} (\dot{v}, \dot{m}, \dot{t}\,)}$$

$$[\textsc{App}] \quad \frac{\begin{array}{c}(e_1, \dot{\sigma}, \dot{m}, \dot{t}\,) \mathrel{\dot{\hookrightarrow}} (\langle \lambda x.e_\lambda, \dot{\sigma}_\lambda \rangle, \dot{m}_\lambda, \dot{t}_\lambda) \\ (e_2, \dot{\sigma}, \dot{m}_\lambda, \dot{t}_\lambda) \mathrel{\dot{\hookrightarrow}} (\dot{v}, \dot{m}_a, \dot{t}_a) \\ (e_\lambda, (x, \dot{\mathsf{tick}}(\dot{t}_a)) :: \dot{\sigma}_\lambda, \dot{m}_a[\dot{\mathsf{tick}}(\dot{t}_a) \mapsto \dot{v}], \dot{\mathsf{tick}}(\dot{t}_a)) \mathrel{\dot{\hookrightarrow}} (\dot{v}', \dot{m}', \dot{t}')\end{array}}{(e_1\ e_2, \dot{\sigma}, \dot{m}, \dot{t}\,) \mathrel{\dot{\hookrightarrow}} (\dot{v}', \dot{m}', \dot{t}')}$$

$$[\textsc{LetE}] \quad \frac{\begin{array}{c}(e_1, \dot{\sigma}, \dot{m}, \dot{t}\,) \mathrel{\dot{\hookrightarrow}} (\dot{v}, \dot{m}', \dot{t}') \\ (e_2, (x, \dot{\mathsf{tick}}(\dot{t}')) :: \dot{\sigma}, \dot{m}'[\dot{\mathsf{tick}}(\dot{t}') \mathrel{\dot{\mapsto}} \dot{v}], \dot{\mathsf{tick}}(\dot{t}')) \mathrel{\dot{\hookrightarrow}} (\dot{\sigma}', \dot{m}'', \dot{t}'')\end{array}}{(\texttt{let}\ x\ e_1\ e_2, \dot{\sigma}, \dot{m}, \dot{t}\,) \mathrel{\dot{\hookrightarrow}} (\dot{\sigma}', \dot{m}'', \dot{t}'')}$$

Abstraction and Concretization

$$\dot{\Sigma} \triangleq \dot{\text{Right}} + \dot{\hookrightarrow} \qquad \text{Trace}^{\#} \triangleq \mathscr{P}(\dot{\Sigma})$$

Definition

Define α : Trace $\rightarrow$ Trace$^{\#}$ and γ : Trace$^{\#}$ $\rightarrow$ Trace by:

$$\alpha(A) \triangleq \{\dot{\alpha}(c) \dot{\hookrightarrow} \dot{\alpha}(r) | c \hookrightarrow r \in A\} \cup \{\dot{\alpha}(r) | r \in A\}$$

$$\gamma(A^{\#}) \triangleq \{c \hookrightarrow r | \dot{\alpha}(c) \dot{\hookrightarrow} \dot{\alpha}(r) \in A^{\#}\} \cup \{r | \dot{\alpha}(r) \in A^{\#}\}$$

Abstract Semantics

Definition (Abstract transfer function)

Given $A^\# \subseteq \dot{\Sigma}$, define:

$$\mathsf{Step}^\#(A^\#) \triangleq \left\{ \dot{c} \mathrel{\dot{\hookrightarrow}} \dot{r}, \dot{r} \,\middle|\, \frac{A'^\#}{\dot{c} \mathrel{\dot{\hookrightarrow}} \dot{r}} \text{ and } A'^\# \subseteq A^\# \text{ and } \dot{c} \in A^\# \right\}$$

Definition (Abstract semantics)

Given $e \in \mathsf{Expr}$ and $S^\# \subseteq \dot{\mathsf{State}}$, define:

$$[\![e]\!]^\# S^\# \triangleq \mathsf{lfp}(\lambda X^\#.\mathsf{Step}^\#(X^\#) \cup \{(e, \dot{s}) | \dot{s} \in S^\#\})$$

Soundness

Lemma (Galois Connection)

$$\mathsf{Trace} = \mathscr{P}(\Sigma) \xleftarrow[\alpha]{\gamma} \mathsf{Trace}^{\#} = \mathscr{P}(\dot{\Sigma})$$

Lemma (Operational Soundness)

For all $c \in \mathsf{Config}$ and $r \in \mathsf{Right}$, $c \hookrightarrow r \Rightarrow \dot{\alpha}(c) \dot{\hookrightarrow} \dot{\alpha}(r)$

Theorem (Soundness)

For all $e \in \mathsf{Expr}$, $\llbracket e \rrbracket \circ \gamma \subseteq \gamma \circ \llbracket e \rrbracket^{\#}$

Finiteness

Theorem (Finiteness)

For all $e \in \text{Expr}$, if $\dot{\mathbb{T}}$ and $S^{\#} \subseteq \dot{\text{State}}$ is finite, $[\![e]\!]^{\#} S^{\#}$ is finite.

Linking Timestamps

$$\dot{\mathbb{T}}_{\infty} \triangleq \dot{\mathbb{T}}_1 + \dot{\mathbb{T}}_2 \quad \dot{\mathsf{tick}}_{\infty}(\dot{t}) \triangleq \begin{cases} \dot{\mathsf{tick}}_1(\dot{t}) & \dot{t} \in \dot{\mathbb{T}}_1 \\ \dot{\mathsf{tick}}_2(\dot{t}) & \dot{t} \in \dot{\mathbb{T}}_2 \end{cases} \quad \dot{\alpha}_{\infty}(t) \triangleq \begin{cases} \dot{\alpha}_1(t) & t \in \mathbb{T}_1 \\ \dot{\alpha}_2(t) & t \in \mathbb{T}_2 \end{cases}$$

Notation

All sets with the subscript $i(i = 1, 2)$ is assumed to be using $\dot{\mathbb{T}}_i$ as timestamps, and all sets with the subscript ∞ is assumed to be using $\dot{\mathbb{T}}_{\infty}$ as timestamps.

Lemma (Injection Preserves $\dot{\hookrightarrow}$)

For all $\dot{s}_1 \in \dot{\mathrm{State}}_1$, $\dot{c}_2 \in \dot{\mathrm{Config}}_2$, $\dot{r} \in \dot{\mathrm{Right}}_2$,

$$\dot{c}_2 \mathrel{\dot{\hookrightarrow}_2} \dot{r}_2 \Rightarrow \dot{c}_2\langle \dot{s}_1 \rangle \mathrel{\dot{\hookrightarrow}_\infty} \dot{r}_2\langle \dot{s}_1 \rangle$$

$$\dot{m}_2\langle \dot{\sigma}_1 \rangle \triangleq \lambda \dot{t}.\{\dot{v}_2\langle \dot{\sigma}_1 \rangle | \dot{v}_2 \in \dot{m}_2(\dot{t})\}$$

Where $\dot{v}_2\langle \dot{\sigma}_1 \rangle$ is the same as concrete injection.

Definition (Abstract Injection)

For $S_1^\# \subseteq \dot{\mathsf{State}}_1$ and $A_2^\# \subseteq \dot{\Sigma}_2$, define:

$$\begin{aligned} S_1^\# \rhd^\# A_2^\# \triangleq & \{\dot{r}_2\langle \dot{s}_1 \rangle | \dot{s}_1 \in S_1^\#, \dot{r}_2 \in A_2^\#\} \cup \\ & \{\dot{c}_2\langle \dot{s}_1 \rangle \dot{\hookrightarrow}_\infty \dot{r}_2\langle \dot{s}_1 \rangle | \dot{s}_1 \in S_1^\#, \dot{c}_2 \dot{\hookrightarrow}_2 \dot{r}_2 \in A_2^\#\} \end{aligned}$$

Definition (Abstract Linking)

For $S_1^\# \subseteq \dot{\mathsf{State}}_1$ and $A_2^\# \subseteq \dot{\Sigma}_2$, define:

$$S_1^\# \propto^\# A_2^\# \triangleq \mathsf{lfp}(\lambda X^\#.\mathsf{Step}_\infty^\#(X^\#) \cup (S_1^\# \rhd^\# A_2^\#))$$

Advance Lemma

Lemma (Abstract Advance)

For all $e \in \mathsf{Expr}$ and $S_1^\# \subseteq \dot{\mathsf{State}}_1$, $S_2^\# \subseteq \dot{\mathsf{State}}_2$,

$$[\![e]\!]^\#(S_1^\# \rhd^\# S_2^\#) = S_1^\# \propto^\# [\![e]\!]^\# S_2^\#$$

Soundness

Corollary (Correctness of $\infty^{\#}$)

For all $e \in \mathsf{Expr}$ and $S_1 \subseteq \mathsf{State}_1$, $S_2 \subseteq \mathsf{State}_2$,

$$S_1 \infty [\![e]\!] S_2 \subseteq \gamma_{\infty}(\alpha_1(S_1) \infty^{\#} [\![e]\!]^{\#} \alpha_2(S_2))$$

Soundness Proof : Modular Analysis + Linking $\cong$ Monolithic Analysis

Access Paths

$$
\begin{array}{rcl}
p & \rightarrow & \epsilon \\
 & \mid & \xrightarrow{x} t\ p \\
 & \mid & \xrightarrow{d} p \\
 & \mid & \xrightarrow{\lambda x.e} p \\
\varphi(\epsilon) & \triangleq & \epsilon \\
\varphi(\xrightarrow{x} t\ p) & \triangleq & \xrightarrow{x} \varphi(t)\ \varphi(p) \\
\varphi(\xrightarrow{M} p) & \triangleq & \xrightarrow{M} \varphi(p) \\
\varphi(\xrightarrow{\lambda x.e} p) & \triangleq & \xrightarrow{\lambda x.e} \varphi(p)
\end{array}
$$

$$\frac{}{\checkmark(\_, m, \epsilon)}$$

$$\frac{t = \sigma(x) \qquad \checkmark(t, m, p)}{\checkmark(\sigma, m, \xrightarrow{x} t\ p)}$$

$$\frac{\sigma' = \sigma(d) \qquad \checkmark(\sigma', m, p)}{\checkmark(\sigma, m, \xrightarrow{d} p)}$$

$$\frac{\langle \lambda x.e, \sigma \rangle = m(t) \qquad \checkmark(\sigma, m, p)}{\checkmark(t, m, \xrightarrow{\lambda x.e} p)}$$

$$\frac{}{\dot{\checkmark}(\_, \dot{m}, \epsilon)}$$

$$\frac{\dot{t} = \dot{\sigma}(x) \qquad \dot{\checkmark}(\dot{t}, \dot{m}, \dot{p})}{\dot{\checkmark}(\dot{\sigma}, \dot{m}, \xrightarrow{x} \dot{t}\ \dot{p})}$$

$$\frac{\dot{\sigma}' = \dot{\sigma}(d) \qquad \dot{\checkmark}(\dot{\sigma}', \dot{m}, \dot{p})}{\dot{\checkmark}(\dot{\sigma}, \dot{m}, \xrightarrow{d} \dot{p})}$$

$$\frac{\langle \lambda x.e, \dot{\sigma} \rangle \in \dot{m}(\dot{t}) \qquad \dot{\checkmark}(\dot{\sigma}, \dot{m}, \dot{p})}{\dot{\checkmark}(\dot{t}, \dot{m}, \xrightarrow{\lambda x.e} \dot{p})}$$

Equivalent Concrete States

Definition (Equivalent Concrete States: $\cong$)

Let $s = (\sigma, m, \_) \in \text{State}$ and $s' = (\sigma', m', \_) \in \text{State}'$. $s \cong s'$ (s is equivalent to s') iff $\exists \varphi \in \mathbb{T} \to \mathbb{T}', \varphi' \in \mathbb{T}' \to \mathbb{T}$:

1. $\forall p \in \text{Path} : \checkmark(\sigma, m, p) \Rightarrow (\checkmark(\sigma', m', \varphi(p)) \wedge p = \varphi'(\varphi(p)))$
2. $\forall p' \in \text{Path}' : \checkmark(\sigma', m', p') \Rightarrow (\checkmark(\sigma, m, \varphi'(p')) \wedge p' = \varphi(\varphi'(p')))$

Concretization Preserves Equivalence

Lemma (Concretization Preserves Equivalence)

Assume that each $\dot{t}, \dot{t'}$ in $\dot{\mathbb{T}}, \dot{\mathbb{T}}'$ corresponds to an infinite set of concrete timestamps. Then for all $S^\# \subseteq \dot{\mathsf{State}}$ and ${S'}^\# \subseteq \dot{\mathsf{State}}'$,

$$S^\# \cong^\# {S'}^\# \Rightarrow \gamma(S^\#) \cong \gamma'({S'}^\#)$$

Evaluation Preserves Equivalence

Lemma (Evaluation Preserves Equivalence)

For all $c \in \text{Config}$, $r \in \text{Right}$, $c' \in \text{Config}'$,

$$c \hookrightarrow r \text{ and } c \cong c' \Rightarrow \exists r' : c' \hookrightarrow r' \text{ and } r \cong r'$$

Thus, if $S \subseteq \text{State}$ and $S' \subseteq \text{State}'$ are equivalent, $[\![e]\!]S \cong [\![e]\!]S'$.

Soundness of Modular Analysis

Given $S^\# \subseteq \dot{\Sigma}$, if $S_1^\# \triangleright^\# S_2^\# \cong^\# S^\#$:

$$\begin{aligned} [\![e]\!]\gamma(S^\#) &\cong [\![e]\!]\gamma_\infty(S_1^\# \triangleright^\# S_2^\#) && (\because \gamma, \hookrightarrow \text{ preserves equivalence}) \\ &\subseteq \gamma_\infty([\![e]\!]^\#(S_1^\# \triangleright^\# S_2^\#)) && (\because \text{Soundness}) \\ &= \gamma_\infty(S_1^\# \infty^\# [\![e]\!]^\# S_2^\#) && (\because \text{Abstract advance}) \end{aligned}$$

Where γ_∞ is derived from:

$\mathbb{T}_\infty \triangleq \mathbb{Z} \times (\dot{\mathbb{T}}_1 + \dot{\mathbb{T}}_2)$ $\quad$ $\mathsf{tick}_\infty(n, \dot{t}) \triangleq (n+1, \dot{\mathsf{tick}}_\infty(\dot{t}))$ $\quad$ $\dot{\alpha}_\infty(n, \dot{t}) \triangleq \dot{t}$

Extension : Parametrized Modules (Functors)

Syntax

$$
\begin{array}{rrcll}
\text{Identifiers} & x, d & \in & \mathsf{Var} & \\
\text{Signature} & s & \in & \mathsf{Sig} & \\
\text{Signature} & s & \rightarrow & [] \mid x :: s \mid (d, s) :: s & \\
\text{Expression} & e & \rightarrow & x \mid \lambda x.e \mid e\ e & \text{untyped } \lambda\text{-calculus} \\
 & & \mid & d \mid \lambda d :> s.e \mid (e\, e) :> s & \text{module calculus, constrained by } s \\
 & & \mid & e \rtimes e & \text{linked expression} \\
 & & \mid & \varepsilon & \text{empty module}
\end{array}
$$

$\boxed{\text{The same theorems hold}}$