프로그램 조각별 따로분석의 이론적 틀

이준협

2023년 11월 3일

ROPAS@SNU

풀고자 한 문제

프로그램 전체를 분석하지 않고, 일부만 미리 분석해놓고 싶다!

```
(* 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
```

- F에 대한 가정만 가지고 분석 (100!+?)
- 이후 M에 대한 정보를 끼워넣음.

풀고자 한 문제

프로그램 전체를 분석하지 않고, 일부만 미리 분석해놓고 싶다!

```
(* 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
```

- F에 대한 가정만 가지고 분석 (100!+?)
- 이후 M에 대한 정보를 끼워넣음.

가정이 부족한 분석?

분석 결과의 재사용?

목표

부족했던 것: **의미구조 정의**부터 자연스럽게 따로분석이 이끌어지는 틀

1. 프로그램의 실행의미가 실제 **실행기**의 동작과 가깝고,
2. 분석 디자이너가 신경 쓸 것이 많이 없는,
3. 그러나 **정밀성**을 임의로 조절할 수 있는 틀, 그리고 안전성 증명.

목차

모듈이 있는 언어

겉모습

Identifiers	x, d	$\in$	Var	
Expression	e	$\rightarrow$	$x \mid \lambda x.e \mid e\ e$	untyped λ-calculus
		$\mid$	$e \rtimes e$	linked expression
		$\mid$	ε	empty module
		$\mid$	d	module identifier
		$\mid$	$\texttt{let}\ x\ e\ e$	expression binding
		$\mid$	$\texttt{let}\ d\ e\ e$	module binding

의미구조 부품들

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

실행 의미구조 (λ 부분)

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

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

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

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

실행 의미구조 (모듈 부분)

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

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

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

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

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

실행 의미구조 (링킹 부분)

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

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

미리 실행시켜도 문제 없다

Lemma (Injection Preserves $\hookrightarrow$)

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

미리 실행시켜도 문제 없다

Lemma (Injection Preserves $\hookrightarrow$)

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

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

일 때.

의미구조끼리 합치기

모듬 의미구조

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

Definition (Transfer function)

Given $A \subseteq \Sigma$, define:

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

Definition (Collecting semantics)

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

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

합치기

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:

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

잘 합쳐진다

Theorem (Advance)

For all $e \in \mathsf{Expr}$ *and* $C_1, C_2 \subseteq \mathsf{Ctx}$,

$$[\![e]\!](C_1 \rhd C_2) = C_1 \propto [\![e]\!]C_2$$

따로분석이란?

$e_1 \triangleq \texttt{let x = 1 in } \varepsilon$

(module $\texttt{M}$)

$e_2 \triangleq \texttt{let fact = fix } \lambda\texttt{fact.}\lambda\texttt{n.if0 n 1 (* n (fact (- n 1))) in } \varepsilon$

(module $\texttt{F}$)

$e \triangleq \texttt{(+ (F} \rtimes \texttt{fact 100) (M} \rtimes \texttt{x))}$

(client code)

$e_\rtimes \triangleq (\texttt{let M = } e_1 \texttt{ in } \varepsilon) \rtimes (\texttt{let F = } e_2 \texttt{ in } \varepsilon) \rtimes e$

(whole program after linking)

따로분석이란?

분석 목표: $e_{\rtimes}$ 실행 중 e가 어떤 환경 아래에서 어떻게 동작할 것인가?

$$\sigma_2 \triangleq [(\mathtt{F}, [(\mathtt{fact}, \langle \text{body of } \mathtt{fact} \rangle)])]$$
$$\sigma_1 \triangleq [(\mathtt{M}, [(\mathtt{x}, 1)])]$$

가정: 적어도 σ_2는 있을 것이다: $[\![e]\!]\{\sigma_2\}$를 근사.

실제: $\underbrace{\{\sigma_2\langle\sigma_1\rangle\}\text{에서 실행되었다}}$

따로분석이란?

분석 목표: $e_{\rtimes}$ 실행 중 e가 어떤 환경 아래에서 어떻게 동작할 것인가?

$$\sigma_2 \triangleq [(\mathtt{F}, [(\mathtt{fact}, \langle \text{body of } \mathtt{fact} \rangle)])]$$
$$\sigma_1 \triangleq [(\mathtt{M}, [(\mathtt{x}, 1)])]$$

가정: 적어도 σ_2는 있을 것이다: $[\![e]\!]\{\sigma_2\}$를 근사.

실제: $\underbrace{\{\sigma_2\langle\sigma_1\rangle\}\text{에서 실행되었다}}_{e_{\rtimes}\text{ 분석하는 중에 } e\text{를 만나면}}$: $\{\sigma_1\} \infty [\![e]\!]\{\sigma_2\}$를 근사.

분석을 위한 의미구조

$\mathbb{T}$와 tick

실행의미는 $\mathbb{T}$(Time)이라는 집합과 tick이라는 함수로 매개화되어있다.

- $\mathbb{T}$: 실행중 프로그램 지점을 구별해줌, 메모리 주소로도 쓰임.
- tick: 현재 환경을 받아서, 증가된(지금껏 안 쓰인) 시간을 줌.

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

분석

요구사항: $\dot{\alpha} : \mathbb{T} \to \dot{\mathbb{T}}$

1. $\dot{\mathsf{tick}} \circ \dot{\alpha} = \dot{\alpha} \circ \mathsf{tick}$인 $\dot{\mathsf{tick}}$ 사용.
2. $\forall \dot{t} : \alpha^{-1}(\dot{t})$는 무한집합.

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

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

따로분석

분석 방법: $[\![e]\!]^{\#} S^{\#}$ 어림잡기

1. 가정($S_2^{\#}$)하고 분석($[\![e]\!]^{\#} S_2^{\#}$)하라
2. 가정이 성립($^{\exists} S_1^{\#} \rhd^{\#} S_2^{\#} \sim^{\#} S^{\#}$)하면, 합쳐라($S_1^{\#} \infty^{\#} [\![e]\!]^{\#} S_2^{\#}$)

감사합니다