프로그램 조각별 따로분석의 이론적 틀

이준협 이광근

서울대학교 프로그래밍 연구실 (ROPAS)

풀고자 한 문제

프로그램 조각이 주어졌을 때, 분석을 해 놓고 기다리자!

- 목표: $e_1 \infty e_2$의 결과 어림잡기.
- 가정: e_1이 무엇을 내보내는지 일부만 알고 있음.
- 질문: e_2를 미리 분석해놓고, e_1을 **따로** 분석해서 **합치면** 전체 결과를 **안전하게** 어림잡을 수 있을까?

따로분석을 위한 의미구조를 엄밀하게 정의해보자!

모듈이 있는 언어의 정의

겉모습 (Untyped λ+Modules)

Identifiers	x, M	$\in$	Var	
Expression	e	$\to$	x	value identifier
		$\mid$	$\lambda x.e$	function
		$\mid$	$e\ e$	application
		$\mid$	$e \infty e$	linked expression
		$\mid$	ε	empty module
		$\mid$	M	module identifier
		$\mid$	$\texttt{let}\ x\ e\ e$	binding expression
		$\mid$	$\texttt{let}\ M\ e\ e$	binding module

속내용 ($\mathbb{T}$, tick에 대해 매개화된)

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

addr(C, x): C의 가장 위에 있는 x의 주소(시간).

ctx(C, M): C의 가장 위에 있는 M의 값(환경).

실행의미 (Operational)

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

$$[\textsc{ExprVar}]\ \frac{t_x = \mathrm{addr}(C, x) \quad v = m(t_x)}{(x, C, m, t) \rightsquigarrow (v, m, t)} \qquad [\textsc{Fn}]\ \frac{}{(\lambda x.e, C, m, t) \rightsquigarrow (\langle \lambda x.e, C \rangle, m, t)}$$

$$[\textsc{AppL}]\ \frac{}{(e_1\ e_2, C, m, t) \rightsquigarrow (e_1, C, m, t)} \qquad [\textsc{AppR}]\ \frac{(e_1, C, m, t) \rightsquigarrow (\langle \lambda x.e_\lambda, C_\lambda \rangle, m_\lambda, t_\lambda)}{(e_1\ e_2, C, m, t) \rightsquigarrow (e_2, C, m_\lambda, t_\lambda)}$$

$$[\textsc{AppBody}]\ \frac{(e_1, C, m, t) \rightsquigarrow (\langle \lambda x.e_\lambda, C_\lambda \rangle, m_\lambda, t_\lambda) \quad (e_2, C, m_\lambda, t_\lambda) \rightsquigarrow (v, m_a, t_a)}{(e_1\ e_2, C, m, t) \rightsquigarrow (e_\lambda, (x, t_a) :: C_\lambda, m_a[t_a \mapsto v], \mathrm{tick}((C, m_a, t_a), x, v))}$$

실행의미 (Collecting)

$$\boxed{\mathrm{Step}(A) \triangleq \left\{ \ell \rightsquigarrow_{\mathrm{tick}} \rho, (\rho, \mathrm{tick}) \,\middle|\, \frac{A'}{\ell \rightsquigarrow_{\mathrm{tick}} \rho} \wedge A' \subseteq A \wedge (\ell, \mathrm{tick}) \in A \right\}}$$

$$\boxed{[\![e]\!]S \triangleq \mathrm{lfp}(\lambda X.\mathrm{Step}(X) \cup \{((e, s), \mathrm{tick}) \mid (s, \mathrm{tick}) \in S\})}$$

모르는 정보가 있어도, 불완전한 증명나무를 저장하고 있다.

"합치기"의 정의

정리 (Linking)

$|[\![e_1]\!]S| \cong S_1 \rhd S_2$임이 확인됐을 때,

$$|[\![e_1 \infty e_2]\!]S| \cong |S_1 \infty [\![e_2]\!]S_2|$$

이다.

- $|[\![e_1]\!]S|$: e_1이 내보내는 환경이
- $S_1 \rhd S_2$: 가정된 S_2와 부족했던 S_1으로 쪼개질 수 있다.
- $|\cdot|$: 최종 상태에 관심이 있다.
- $\cong$: "동일한 이름들을 내보내는" 의미구조면 된다.

$\cong$의 정의

C, m : 이름(x, M)으로만 접근 가능한 구조들.

$$\begin{aligned} \mathrm{Step}_m(G) &\triangleq \{C \xrightarrow{x} t, t \mid C \in G \wedge t = \mathrm{addr}(C, x)\} \\ &\cup \{C \xrightarrow{M} C', C' \mid C \in G \wedge C' = \mathrm{ctx}(C, M)\} \\ &\cup \{t \xrightarrow{e} C, C \mid t \in G \wedge \langle e, C \rangle = m(t)\} \\ \underline{C, m} &\triangleq \mathrm{lfp}(\lambda X.\mathrm{Step}_m(X) \cup \{C\}) \end{aligned}$$

$\underline{C, m}$: 이름으로 뽑아낼 수 있는 모든 정보를 담은 rooted graph.

$(C_1, m_1) \cong (C_2, m_2) \triangleq \underline{C_1, m_1} \cong \underline{C_2, m_2}$: 그래프가 동형

$\rhd$, ∞의 정의

$(s_1, \mathrm{tick}_1) \rhd (s_2, \mathrm{tick}_2) \triangleq (s_+, \mathrm{tick}_+)$

- s_+는 $s_1 = (C_1, m_1, t_1)$을 $s_2 = (C_2, m_2, t_2)$에 **끼워넣은** 것.
- tick_+는 $t \in \mathbb{T}_1 + \mathbb{T}_2$를 $t \in \mathbb{T}_1$의 경우 tick_1으로, $t \in \mathbb{T}_2$의 경우 **끼워넣기 전** 상태를 바탕으로 tick_2로 증가시키는 함수.

$$\boxed{S \infty A \triangleq \mathrm{lfp}(\underbrace{\lambda X.\mathrm{Step}(X) \cup \overbrace{(S \rhd A)}^{\text{끼워넣고}}}_{\text{나머지를 채운다}})}$$

도움정리 (Advance)

따로따로 나눌 수 있는 곳에서 출발하면, 미리 하고 합친 것과 같다.

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

분석을 위해서

요구사항: $\alpha : \mathbb{T} \to \mathbb{T}^\#$

1. $\mathrm{tick}^\# \circ \alpha = \alpha \circ \mathrm{tick}$인 $\mathrm{tick}^\#$ 사용.
2. $\forall t^\# : \alpha^{-1}(t^\#)$는 $\mathbb{T}$의 순서에 대해 윗뚜껑이 없다.

분석 방법: $|[\![e_1 \infty e_2]\!]S^\#|$ 어림잡기

1. 가정$(S_2^\#)$하고 분석$([\![e_2]\!]^\# S_2^\#)$하라
2. 가정이 성립$(S_1^\# \rhd^\# S_2^\# \cong^\# |[\![e_1]\!]^\# S^\#|)$하면, 합쳐라$(S_1^\# \infty^\# [\![e_2]\!]^\# S_2^\#)$

`github.com/LimitEpsilon/modular-analysis`