프로그램 따로분석의 이론적 기틀: 재귀적 모듈 지원하기

이준협 이광근

서울대학교 프로그래밍 연구실 (ROPAS)

문제 소개

무엇을 하고 싶은가?

- 프로그램 외부의 값을 몰라도 분석하고 싶다.
  ex. 외부 라이브러리 함수를 부르는 경우
- 이름이 “복잡한 방식”으로 알려질 때에도 분석하고 싶다.
  ex. First-class modules, Functors, Recursive modules

어떻게 할 것인가?

- “Algebraic Effects”: 환경을 받는 **함수**를 **자료구조**로 나타냄.
- 환경과의 상호작용을 요약하고, 나중에 환경이 들어오면 풀자!

모듈이 있는 언어의 정의

겉모습 (Untyped λ+Modules)

$$
\begin{array}{rrcll}
\text{Identifiers} & x & \in & \text{Var} & \\
\text{Expression} & e & \to & x \mid \lambda x.e \mid e\ e & \lambda\text{-calculus} \\
 & & \mid & e \rtimes e & \text{linked expression} \\
 & & \mid & \varepsilon & \text{empty module} \\
 & & \mid & x = e\,;e & \text{(recursive) binding}
\end{array}
$$

속내용

$$
\begin{array}{rrcll}
\text{Environment} & \sigma & \in & \text{Env} & \\
\text{Location} & \ell & \in & \text{Loc} & \\
\text{Value} & v & \in & \text{Val} \triangleq \text{Env} + \text{Var} \times \text{Expr} \times \text{Env} & \\
\text{Weak Value} & w & \in & \text{WVal} \triangleq \text{Val} + \text{Loc} \times \text{Val} & \\
\text{Environment} & \sigma & \to & \bullet & \text{empty stack} \\
 & & \mid & (x, w) :: \sigma & \text{weak value binding} \\
 & & \mid & (x, \ell) :: \sigma & \text{free location binding} \\
\text{Value} & v & \to & \sigma & \text{exported environment} \\
 & & \mid & \langle \lambda x.e, \sigma \rangle & \text{closure} \\
\text{Weak Value} & w & \to & v & \text{value} \\
 & & \mid & \mu\ell.v & \text{recursive value}
\end{array}
$$

실행의미

$\boxed{(e, \sigma) \Downarrow v}$

$$
\textsc{Id}\ \frac{\sigma(x) = v}{(x, \sigma) \Downarrow v}
\qquad
\boxed{\textsc{RecId}\ \frac{\sigma(x) = \mu\ell.v}{(x, \sigma) \Downarrow v[\mu\ell.v/\ell]}}
\qquad
\textsc{Fn}\ \frac{}{(\lambda x.e, \sigma) \Downarrow \langle \lambda x.e, \sigma \rangle}
$$

$$
\textsc{App}\ \frac{(e_1, \sigma) \Downarrow \langle \lambda x.e, \sigma_1 \rangle \quad (e_2, \sigma) \Downarrow v_2 \quad (e, (x, v_2) :: \sigma_1) \Downarrow v}{(e_1\ e_2, \sigma) \Downarrow v}
$$

$$
\boxed{\textsc{Link}\ \frac{(e_1, \sigma) \Downarrow \sigma_1 \quad (e_2, \sigma_1) \Downarrow v}{(e_1 \rtimes e_2, \sigma) \Downarrow v}}
\qquad
\textsc{Empty}\ \frac{}{(\varepsilon, \sigma) \Downarrow \bullet}
$$

$$
\boxed{\textsc{RecBind}\ \frac{\ell \notin \text{FLoc}(\sigma) \quad (e_1, (x, \ell) :: \sigma) \Downarrow v_1 \quad (e_2, (x, \mu\ell.v_1) :: \sigma) \Downarrow \sigma_2}{(x = e_1; e_2, \sigma) \Downarrow (x, \mu\ell.v_1) :: \sigma_2}}
$$

상호작용 기록하고 나중에 풀기

상호작용 기록

$$
\begin{array}{rrcll}
\text{Environment} & \sigma & \to & \cdots & \\
 & & \mid & [E] & \text{answer to an event} \\
\text{Value} & v & \to & \cdots & \\
 & & \mid & E & \text{answer to an event} \\
\text{Event} & E & \to & \text{Init} & \text{initial environment} \\
 & & \mid & \text{Read}(E, x) & \text{read event} \\
 & & \mid & \text{Call}(E, v) & \text{call event}
\end{array}
$$

$$
[E](x) \triangleq \text{Read}(E, x)
\qquad
\textsc{AppEvent}\ \frac{(e_1, \sigma) \Downarrow E \quad (e_2, \sigma) \Downarrow v}{(e_1\ e_2, \sigma) \Downarrow \text{Call}(E, v)}
$$

상호작용 풀기 (일부 규칙들)

$\boxed{w \downarrow_{\sigma_0} W}$

$$
\textsc{R-Init}\ \frac{}{\text{Init} \downarrow \sigma_0}
\qquad
\textsc{R-Read}\ \frac{E \downarrow \Sigma \quad \Sigma(x) = V}{\text{Read}(E, x) \downarrow V}
\qquad
\textsc{R-ReadRec}\ \frac{E \downarrow \Sigma \quad \Sigma(x) = \mu\ell.V}{\text{Read}(E, x) \downarrow V[\mu\ell.V/\ell]}
$$

$$
\textsc{R-CallV}\ \frac{E \downarrow \langle \lambda x.e, \Sigma \rangle \quad v \downarrow V \quad (e, (x, V) :: \Sigma) \Downarrow V'}{\text{Call}(E, v) \downarrow V'}
\qquad
\textsc{R-CallE}\ \frac{E \downarrow E' \quad v \downarrow V}{\text{Call}(E, v) \downarrow \text{Call}(E', V)}
$$

Conjecture (상호작용 풀기는 안전하다)

$$
\sigma \downarrow_{\sigma_0} \Sigma \text{ and } (e, \sigma) \Downarrow v \text{ and } (e, \Sigma) \Downarrow V \Rightarrow v \downarrow_{\sigma_0} V
$$

예시

```
(Top= (Tree=(max=λx.f (Top⋊Forest⋊max) x; ε);
       Forest=(max=λx.(Top⋊Tree⋊max) x; ε);
       ε);
    ε) ⋊((Top⋊Tree⋊max) id)
```

$\Downarrow$ Call(Call(Read(Init, “f”), $\langle\lambda$`x.Top` $\rtimes$ `Tree` $\rtimes$ `max x`$, \ldots\rangle$), `id`)

요약하기

메모리의 등장

- 메모리가 있으면 요약이 편해진다.
- $\text{Env} \triangleq \text{Var} \xrightarrow{\text{fin}} \text{Loc}$, $\text{Mem} \triangleq \text{Loc} \xrightarrow{\text{fin}} \text{Val}$이면, Loc만 요약.
- 그렇다면 실행의미가 동일하다는 것부터 보여야 한다.

$$
\sigma \sim \sigma_0, m_0 \wedge (e, \sigma) \Downarrow v \Rightarrow \exists v_1, m_1 \sim v : (e, \sigma_0, m_0) \Downarrow (v_1, m_1)
$$
$$
\sigma \sim \sigma_0, m_0 \wedge (e, \sigma_0, m_0) \Downarrow (v_1, m_1) \Rightarrow \exists v \sim v_1, m_1 : (e, \sigma) \Downarrow v
$$

$\Downarrow$와 $\downarrow_{\sigma_0}$의 요약 (진행 중)

- 첫번째: $\downarrow$를 메모리가 있을 때 “안전”하게 정의하기
  “안전”: $w \sim v, m \wedge w \downarrow W \Rightarrow \exists V, M : W \sim V, M \wedge v, m \downarrow V, M$
- 두번째: 메모리의 요약에 대해 “안전”하게 $\downarrow$, $\Downarrow$를 요약