



# 프로그램 조각별 따로분석의 이론적 틀

---

이준협

2023년 8월 23일

SIGPL 2023 여름학교

## 풀고자 한 문제

프로그램 전체를 분석하지 않고, 일부만 미리 분석해놓고 싶다!

- 상황 1: 외부 모듈에 대한 가정 없이 분석 후 재사용
- 상황 2: 예전에 다른 모듈과 합쳐서 분석했던 결과를 재사용

## 풀고자 한 문제

프로그램 전체를 분석하지 않고, 일부만 미리 분석해놓고 싶다!

- 상황 1: 외부 모듈에 대한 가정 없이 분석 후 재사용
- 상황 2: 예전에 다른 모듈과 합쳐서 분석했던 결과를 재사용

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

주요 정리

분석을 위해 신경 쓸 것

## 모듈이 있는 언어

---

## 겉모습

Identifiers	$x, M$	$\in$	Var	
Expression	$e$	$\rightarrow$	$x$	value identifier
			$\lambda x.e$	function
			$e\ e$	application
			$e \propto e$	<u>linked expression</u>
			$\varepsilon$	empty module
			$M$	<u>module identifier</u>
			$\text{let } x\ e\ e$	binding expression
			$\text{let } M\ e\ e$	binding module

# 예시

```
(* 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)))
```



## 주요 정리

---

## 따로분석이란?

최종목표:  $S$ 에서 출발한  $e_1 \times e_2$  의 결과 :  $\llbracket e_1 \times e_2 \rrbracket S$

## 따로분석이란?

최종목표:  $S$ 에서 출발한  $e_1 \times e_2$ 의 결과 :  $|\llbracket e_1 \times e_2 \rrbracket S|$

- 원래는, 먼저  $S$ 에서 출발해  $e_1$ 의 결과를 계산 :  $|\llbracket e_1 \rrbracket S|$
- $e_1$ 의 결과에서 출발해  $e_2$ 의 결과를 계산 :  $|\llbracket e_2 \rrbracket \llbracket e_1 \rrbracket S|$

$$|\llbracket e_1 \times e_2 \rrbracket S| = |\llbracket e_2 \rrbracket \llbracket e_1 \rrbracket S|$$

## 따로분석이란?

최종목표:  $S$ 에서 출발한  $e_1 \times e_2$ 의 결과:  $|\llbracket e_1 \times e_2 \rrbracket S|$

- 원래는, 먼저  $S$ 에서 출발해  $e_1$ 의 결과를 계산:  $|\llbracket e_1 \rrbracket S|$
- $e_1$ 의 결과에서 출발해  $e_2$ 의 결과를 계산:  $|\llbracket e_2 \rrbracket \llbracket e_1 \rrbracket S|$

$$|\llbracket e_1 \times e_2 \rrbracket S| = |\llbracket e_2 \rrbracket \llbracket e_1 \rrbracket S|$$

- 대신, 가정된  $S_2$ 에서 출발해  $e_2$ 을 가능한 곳까지 계산:  $|\llbracket e_2 \rrbracket S_2|$
- 이후,  $e_1$ 의 결과가  $S_2$ 와 나머지로 분리되나 확인:  $|\llbracket e_1 \rrbracket S| \cong S_1 \triangleright S_2$
- 부족했던 부분인  $S_1$ 을 합쳐서 최종 결과 계산:  $S_1 \times |\llbracket e_2 \rrbracket S_2|$

$$|\llbracket e_1 \times e_2 \rrbracket S| \cong |S_1 \times |\llbracket e_2 \rrbracket S_2||$$

분석을 위해 신경 쓸 것

---

## T와 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.

## 분석

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

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

분석 방법:

1. 가정( $S_2^\#$ )하고 분석( $(\llbracket e_2 \rrbracket^\# S_2^\#)$ )하라
2. 가정이 성립( $S_1^\# \triangleright^\# S_2^\# \cong^\# (\llbracket e_1 \rrbracket^\# S^\#)$ )하면, 합쳐라( $S_1^\# \bowtie^\# (\llbracket e_2 \rrbracket^\# S_2^\#)$ )

## 홍보

포스터에 언어의 실행의미 등 더 자세한 설명이 적혀있습니다!