

전자기기 사용 금지

Here's What Happened When I Made My College Students Put Away Their Phones

Aug. 31, 2025



D 한국경제

내년 3월부터 수업 중 스마트폰 쓰면 '불법'...학생들 '멘붕'

이슬기 · 2025. 8. 27. 16:21

🔊 🔊 🔊 🔊 🔊



Computers & Education

Volume 62, March 2013, Pages 24-31



Laptop multitasking hinders classroom learning for both users and nearby peers

Faria Sana ^a, Tina Weston ^{b,c}, Nicholas J. Cepeda ^{b,c}

Show more

Outline | Share | Cite

<https://doi.org/10.1016/j.compedu.2012.10.003>

[Get rights and content](#)

Under a Creative Commons [license](#)

[Open access](#)

Abstract

Laptops are commonplace in university classrooms. In light of cognitive psychology theory on costs associated with multitasking, we examined the effects of in-class laptop use on student learning in a simulated classroom. We found that

SNU 프로그래밍언어 특강

(0.1 + 1.0)

이 광근

kwangkeunyi.snu.ac.kr

계획

- ▶ 준비
 - ▶ 집합_{set}, 함수_{function}, 관계_{relation}
 - ▶ 합집합, 곱집합, 함수집합, 유한함수집합
 - ▶ PO_{partial order}, 체인_{chain}, $\sqcup$, $\sqcap$, $\perp$
 - ▶ CPO_{complete partial order}, 합CPO, 곱CPO, 함수CPO
 - ▶ 연속함수_{continuous function}, 최소고정점_{least fixpoint}
- ▶ 문법구조_{syntax}, 의미구조_{semantics}
- ▶ 무엇의미구조_{denotational semantics}

집합, 관계

집합 $S \rightarrow \Phi$

인덱스로, 코덱으로

$$\begin{array}{l} | \{\cdots\} \mid \{x \mid x \text{의 조건}\} \\ | S + S \mid S \times S \\ | S \rightarrow S \mid S \xrightarrow{\text{fin}} S \\ | 2^S \end{array}$$

- ▶ $A + B = \{\langle a, 1 \rangle \mid a \in A\} \cup \{\langle b, 2 \rangle \mid b \in B\}.$
- ▶ $A \times B = \{\langle a, b \rangle \mid a \in A, b \in B\}.$
- ▶ $A \rightarrow B = \{\text{함수 } f \mid \text{domain}(f) = A, \text{range}(f) \subseteq B\}$
- ▶ $A \xrightarrow{\text{fin}} B = \{\text{함수 } f \mid \text{domain}(f) \overset{\text{fin}}{\subseteq} A, \text{range}(f) \subseteq B\}$
- ▶ $2^A = \{X \mid X \subseteq A\}$

부분순서 partial order

- ▶ 순서 $\sqsubseteq$
 - ▶ 거울_{reflexive}: $x \sqsubseteq x$
 - ▶ 한방향_{anti-symmetric}: $x \sqsubseteq y \wedge y \sqsubseteq x$ 이면 $x = y$
 - ▶ 번짐_{transitive}: $x \sqsubseteq y \wedge y \sqsubseteq z$ 이면 $x \sqsubseteq z$
- ▶ 부분순서집합 S : S 의 (일부) 원소들 사이에 순서 $\sqsubseteq$ 가 있으면
- ▶ 체인_{chain}: 순서대로 일렬로 줄 서는 집합(모든 원소들 사이에 순서가 있는 집합)
- ▶ $\sqcup X, X \subseteq S$: 최소윗뚜껑_{least upper bound} (존재안할수도)
- ▶ $\sqcap X, X \subseteq S$: 최대아래뚜껑_{greatest lower bound} (존재안할수도)

프로그램 집합 = 인덱스로 정의

프로그램의 생김새

- ▶ 나무구조를 갖춘 2차원 모습

정수식

$$\begin{array}{lcl} E & \rightarrow & n \quad (n \in \mathbb{Z}) \\ & | & E + E \\ & | & -E \end{array}$$

명령형 언어 프로그램

$$\begin{array}{l} C \rightarrow \text{skip} \\ | \quad x := E \\ | \quad \text{if } E \ C \ C \\ | \quad C ; C \end{array}$$

군더더기 없이

$$\begin{array}{lcl} C & \rightarrow & \& \\ & | & = x \ E \\ & | & ? \ E \ C \ C \\ & | & ; \ C \ C \end{array}$$

$$\begin{array}{lcl} E & \rightarrow & n \quad (n \in \mathbb{Z}) \\ & | & + \ E \ E \\ & | & - \ E \end{array}$$

$$\begin{array}{lcl}
 C & \rightarrow & \& \\
 | & & x\ E \\
 | & & E\ C\ C \\
 | & & C\ C
 \end{array}$$

$$\begin{array}{lcl}
 E & \rightarrow & n \qquad (n \in \mathbb{Z}) \\
 | & & E\ E \\
 | & & -\ E
 \end{array}$$

이렇게 까지 최대한 핵심만?

요약된 v.s. 구체적인 문법구조

- ▶ 요약된 문법구조_{abstract syntax}
 - ▶ 프로그램을 만들 때 사용하는 규칙
 - ▶ 나무구조를 가진 2차원의 구조물
- ▶ 구체적인 문법구조_{concrete syntax}
 - ▶ 읽을 때 사용하는 규칙
 - ▶ 프로그램의 표현: 1차원의 실
 - ▶ 1차원의 실에서 2차원의 구조물을 복구하는 규칙

구체적인 문법 concrete syntax

$-1+2$ 는 $((-1) + 2)$? $-(1 + 2)$?

▶ 답변 불가

$$\begin{array}{lcl} E & \rightarrow & n \quad (n \in \mathbb{Z}) \\ & | & E + E \\ & | & - E \end{array}$$

▶ 답변 가능

$$\begin{array}{lcl} E & \rightarrow & n \quad (n \in \mathbb{Z}) \\ & | & E + E \\ & | & - F \\ F & \rightarrow & n \\ & | & (E) \end{array}$$

구체적인 문법 concrete syntax

- ▶ 1차원의 실에서 2차원의 구조물을 혼동없이 복구시키는 규칙
 - ▶ 방향성 associativity 과 우선순위 precedence
- ▶ 프로그램 복원 parsing 혹은 문법검증 parsing 과정의 설계도

이제부터 “프로그램” 하면

요약된 문법 `abstract syntax`으로 만들어진 2차원의 나무 구조물

- ▶ 편의를 위해서 1차원 실로 표현
- ▶ 적절히 괄호를 이용해서 그 구조를 명시

의미구조_{semantics}

- ▶ 프로그램이 뜻하는 바를 정의
- ▶ 프로그램이 뜻 하는 바?

“1+2”라는 프로그램의 뜻?

- ▶ 의미 = 단도직입/무엇: “3”
 뭐냐를 드러내는, 무엇 의미구조_{denotational semantics}
- ▶ 의미 = 계산과정/어떻게: “1과 2를 더해 3을 냄”
 과정을 드러내는, 실행 의미구조_{operational semantics}

다양하다

프로그램의 의미를 혼동없이 표현하는 스타일들

- ▶ 모두 충분히 엄밀
- ▶ 각 스타일마다 다양한 증명 기술이 존재
- ▶ 각 스타일이 어울리는 경우가 있음
- ▶ 우리 의도에 맞는 의미구조 방식을 선택

무엇 의미구조_{denotational semantics}

- ▶ 프로그램의 의미: 전통적인 수학의 세계에서, 의미하는 바 무엇인지를 드러냄
- ▶ 단도직입, 결국 무엇인지를 드러내는 의미구조
- ▶ (별명) 조립식 의미구조_{compositional semantics}:
전체의 의미 = 부품들의 의미로 조립
- ▶ (별명) 고정점 의미구조_{fixpoint semantics}:
함수의 고정점으로 정의

$$\begin{array}{lcl}
C & \rightarrow & \text{skip} \\
& | & x := E \\
& | & \text{if } E \ C \ C \\
& | & C ; C \\
E & \rightarrow & n \qquad (n \in \mathbb{Z}) \\
& | & x \\
& | & E + E \\
& | & - E
\end{array}$$

- ▶ 명령어 C 의 의미 = 함수: 메모리에서 메모리로
- ▶ 메모리 = 함수: 주소에서 값으로

의미공간 semantic domain

의미를 정의할 때 사용하는 물건들의 집합

$$M \in Memory = Var \rightarrow Value$$

$$z \in Value = \mathbb{Z}$$

$$x \in Var = ProgramVariable$$

$$\text{명령문 } C \text{의 의미 } \llbracket C \rrbracket \in Memory \rightarrow Memory$$

$$\text{계산식 } E \text{의 의미 } \llbracket E \rrbracket \in Memory \rightarrow \mathbb{Z}$$

$$\begin{aligned}
\llbracket \text{skip} \rrbracket M &= M \\
\llbracket x := E \rrbracket M &= M\{x \mapsto \llbracket E \rrbracket M\} \\
\llbracket \text{if } E \ C_1 \ C_2 \rrbracket M &= \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } \llbracket C_1 \rrbracket M \text{ else } \llbracket C_2 \rrbracket M \\
\llbracket C_1 ; C_2 \rrbracket M &= \llbracket C_2 \rrbracket (\llbracket C_1 \rrbracket M) \\
\llbracket n \rrbracket M &= n \\
\llbracket x \rrbracket M &= M(x) \\
\llbracket E_1 + E_2 \rrbracket M &= (\llbracket E_1 \rrbracket M) + (\llbracket E_2 \rrbracket M) \\
\llbracket - E \rrbracket M &= -(\llbracket E \rrbracket M)
\end{aligned}$$

잘 정의되었는가? 조립식인가?

조립식이 안되는 경우

`while E C`

의 의미

$$\begin{aligned} & \llbracket \text{while } E \ C \rrbracket M \\ &= \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } \llbracket \text{while } E \ C \rrbracket (\llbracket C \rrbracket M) \text{ else } M \end{aligned}$$

조립식인가?

$$\llbracket \text{while } E \ C \rrbracket M$$

$$= \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } \llbracket \text{while } E \ C \rrbracket (\llbracket C \rrbracket M) \text{ else } M$$

- ▶ 정의가 아님; 방정식일 뿐
- ▶ 그 해는 무엇일까? 해는 과연 있을까? 항상 있을까?
있다면 유일하게 있을까?

- ▶ 그 질문들에 대한 답은 모두 “예”
- ▶ 의미방정식에서 사용하는 물건들이 소속된 공간_{semantic domain}이 특별하기 때문
- ▶ 읽기: [Sco89,Sco72,Sco70]

모든 컴퓨터 프로그램(모든 계산 가능한 함수들)의 의미 (의미방정식의 해)는 의미공간 이론에서 규정하는 성질의 집합안에서 유일하게 존재하고 그것은 이리이러하다.

의미공간 이론 domain theory

- ▶ 의미공간 = CPO complete partial order set
- ▶ 프로그램의 의미방정식들에 쓰이는 것들은 모두 어떤 CPO의 원소들
- ▶ 연산자들은 모두 CPO에서 CPO로 가는 연속함수 continuous function
 - ▶ “Computability is continuity.”
 - ▶ “프로그램은 연속함수이다.”

프로그램 C 의 의미 $\llbracket C \rrbracket$ 에 대한 의미방정식은 항상 다음과 같고

$$\llbracket C \rrbracket = \mathcal{F}(\llbracket C \rrbracket)$$

여기서 $\llbracket C \rrbracket \in D$ (어떤 CPO)

그리고 $\mathcal{F} \in D \rightarrow D$ (D 에서 D 로의 연속함수들의 CPO)

- ▶ 이 방정식의 해는 항상 연속함수 $\mathcal{F}$ 의 최소고정점 least fixpoint으로 정의.
- ▶ 고정점 의미구조 fixpoint semantics

CPO (의미공간)

프로그램의 의미는 CPO(*complete partial order*)라는 공간의 한 원소

- ▶ CPO 는 집합
- ▶ 집합의 원소들 간에 어떤 순서가 있고(*partial order*)
- ▶ 모든 원소보다작은 밑바닥 원소($\perp$)가 있고
- ▶ 그 순서로 일렬로 줄 세울 수 있는 원소들(*chain*)의 최소윗뚜껑least upper bound, LUB이 항상 그 집합안에 있다.

연속함수_{continuous function}, 최소고정점_{least fixpoint}

- ▶ CPO A , CPO B . 연속함수 $f : A \rightarrow B$ 란, A 의 체인_{chain}의 최소위뚜껑_{least upper bound}를 보존해주는 함수:

$$\forall \text{체인 } X \subseteq A. f(\bigsqcup X) = \bigsqcup_{x \in X} f(x).$$

- ▶ 연속함수 $f : A \rightarrow A$ 의 최소고정점 $\text{lfp } f$ 은

$$\text{lfp } f = \bigsqcup_{i \in \mathbb{N}} f^i(\perp) = \bigcap \{x \mid f(x) \sqsubseteq x\}.$$

- ▶ 연속함수는 단조함수
- ▶ 단조함수 $f: x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$
- ▶ 팽창함수 $f: x \sqsubseteq f(x)$

소풍: 일반적으로, 고정점_{fixpoint} 예

“Computer science is full of fixpoints.”

소풍: 일반적으로, 고정점_{fixpoint} 예

“Computer science is full of fixpoints.”

$$\begin{aligned}\blacktriangleright N &= \{0\} \cup \{n+1 \mid n \in N\} \\ &= lfp \lambda X. \{0\} \cup \{n+1 \mid n \in X\}\end{aligned}$$

소품: 일반적으로, 고정점_{fixpoint} 예

“Computer science is full of fixpoints.”

- ▶ $N = \{0\} \cup \{n+1 \mid n \in N\}$
 $= \text{lfp} \lambda X. \{0\} \cup \{n+1 \mid n \in X\}$
- ▶ $\text{list} = \{\text{nil}\} \cup \{(0, l) \mid l \in \text{list}\}$
 $= \text{lfp} \lambda X. \{\text{nil}\} \cup \{(0, l) \mid l \in X\}$

소품: 일반적으로, 고정점_{fixpoint} 예

“Computer science is full of fixpoints.”

- ▶ $N = \{0\} \cup \{n+1 \mid n \in N\}$
 $= \text{lfp} \lambda X. \{0\} \cup \{n+1 \mid n \in X\}$
- ▶ $\text{list} = \{\text{nil}\} \cup \{(0, l) \mid l \in \text{list}\}$
 $= \text{lfp} \lambda X. \{\text{nil}\} \cup \{(0, l) \mid l \in X\}$
- ▶ $\text{reach}(N) = N \cup \text{reach}(\text{next}(N))$
 $= \text{lfp} \lambda f. (\lambda N. N \cup f(\text{next}(N)))$

소풍: 일반적으로, 고정점_{fixpoint} 예

“Computer science is full of fixpoints.”

- ▶ $N = \{0\} \cup \{n+1 \mid n \in N\}$
 $= \text{lfp} \lambda X. \{0\} \cup \{n+1 \mid n \in X\}$
- ▶ $\text{list} = \{\text{nil}\} \cup \{(0, l) \mid l \in \text{list}\}$
 $= \text{lfp} \lambda X. \{\text{nil}\} \cup \{(0, l) \mid l \in X\}$
- ▶ $\text{reach}(N) = N \cup \text{reach}(\text{next}(N))$
 $= \text{lfp} \lambda f. (\lambda N. N \cup f(\text{next}(N)))$
- ▶ $\text{sort}(A) = \text{if sorted}(A) \text{ then } A \text{ else } \text{sort}(\text{exch}(A))$
 $= \text{lfp} \lambda f. (\lambda A. \text{sorted}(A) ? A : f(\text{exch}(A)))$

소품: 일반적으로, 고정점_{fixpoint} 예

“Computer science is full of fixpoints.”

- ▶ $N = \{0\} \cup \{n+1 \mid n \in N\}$
 $= \text{lfp} \lambda X. \{0\} \cup \{n+1 \mid n \in X\}$
- ▶ $\text{list} = \{\text{nil}\} \cup \{(0, l) \mid l \in \text{list}\}$
 $= \text{lfp} \lambda X. \{\text{nil}\} \cup \{(0, l) \mid l \in X\}$
- ▶ $\text{reach}(N) = N \cup \text{reach}(\text{next}(N))$
 $= \text{lfp} \lambda f. (\lambda N. N \cup f(\text{next}(N)))$
- ▶ $\text{sort}(A) = \text{if sorted}(A) \text{ then } A \text{ else } \text{sort}(\text{exch}(A))$
 $= \text{lfp} \lambda f. (\lambda A. \text{sorted}(A) ? A : f(\text{exch}(A)))$
- ▶ $\text{fac}(n) = \text{if } n=0 \text{ then } 1 \text{ else } n * \text{fac}(n-1)$
 $= \text{lfp} \lambda f. (\lambda n. n = 0 ? 1 : n \times f(n-1))$

CPO 만들기

$$\begin{array}{rcl} \text{집합} & S & \\ \text{CPO} & D & \rightarrow S_{\perp} \\ & | & D + D \\ & | & D \times D \\ & | & D \rightarrow D \end{array}$$

- ▶ 올려붙인 집합 $\text{lifted set } S_{\perp}$ 은 CPO
- ▶ CPO와 CPO의 곱 product 도 CPO
- ▶ CPO와 CPO의 합 sum 도 CPO
- ▶ CPO에서 CPO로 가는 연속함수들의 집합도 CPO

$$S_{\perp} = S \cup \{\perp\}$$

- ▶ S 원소들 사이의 순서는 없고
- ▶ 순서는 오직 $\perp$ 과 S 사이에만: $\forall x \in S. \perp \sqsubseteq x$.
- ▶ 모든 체인은 유한. 따라서 CPO.

CPO D_1 과 D_2 의 데카르트 곱 Cartesian product

$$D_1 \times D_2 = \{\langle x, y \rangle \mid x \in D_1, y \in D_2\}$$

- ▶ 원소들의 순서는 조립식 component-wise

$$\langle x, y \rangle \sqsubseteq \langle x', y' \rangle \text{ iff } x \sqsubseteq_{D_1} x' \wedge y \sqsubseteq_{D_2} y'.$$

- ▶ 따라서 CPO. (왜?)

CPO D_1 과 D_2 의 합

$$D_1 + D_2 = \{\langle x, 1 \rangle \mid x \in D_1\} \cup \{\langle x, 2 \rangle \mid x \in D_2\} \cup \{\perp\}$$

▶ 원소들의 순서는 출신별로

$$\langle x, 1 \rangle \sqsubseteq \langle x', 1 \rangle \quad \text{iff} \quad x \sqsubseteq_{D_1} x'$$

$$\langle x, 2 \rangle \sqsubseteq \langle x', 2 \rangle \quad \text{iff} \quad x \sqsubseteq_{D_2} x'$$

▶ 따라서 CPO. (왜?)

CPO D_1 에서 D_2 로 가는 모든 연속함수의 집합 $D_1 \rightarrow D_2$ 은 CPO

- ▶ 연속함수들 순서는 조립식_{point-wise}:

$$f \sqsubseteq g \text{ iff } \forall x \in D_1. f(x) \sqsubseteq_{D_2} g(x).$$

- ▶ 따라서 CPO. (왜?)

의미구조에서 연속함수 표기법

$$\lambda x. \dots x \dots$$

(x 는 함수의 인자).

- ▶ 1을 더하는 연속함수: $\lambda x. x + 1$
- ▶ 함수 f 를 2에 적용: $f2, f(2)$

CPO는 모든 프로그래밍언어의 의미공간으로 충분

CPO사이에서 위의 방법으로 정의된 CPO 방정식(domain equation)의 해가 항상 존재.

▶ 예:

$$Store = Loc \rightarrow (Int + Loc + Cmd)$$

$$Cmd = Store \rightarrow Store$$

를 만족하는 CPO $Store$ 는 항상 존재

▶ 참고:

$$D = D \rightarrow D$$

를 만족하는 CPO D 도 존재(Dana Scott's domain theory). 이 사실이, 왜 CPO가 모든 프로그램의 의미를 담을 수 있는 그릇인지를 말해준다. (왜?)

- ▶ 의미방정식에서 사용하는 모든 물건들은 CPO의 원소
- ▶ 연산자들은 모두 CPO에서 CPO로 가는 연속함수들
- ▶ 따라서 모든 의미방정식의 해는 항상 어떤 연속함수 $\mathcal{F}$ 의 고정점:

$$X = \mathcal{F}(X)$$

- ▶ 위의 방정식의 해는 $\mathcal{F}$ 의 최소고정점 $lfp\mathcal{F}$ 로 정의:

$$lfp\mathcal{F} = \sqcup_{i \in \mathbb{N}} \mathcal{F}^i(\perp)$$

그래서 while-문의 의미는?

$M \in Memory = Var \rightarrow Value$ 연속함수 CPO

$z \in Value = \mathbb{Z}_\perp$ 올려붙인 CPO

$x \in Var = ProgramVariable_\perp$ 올려붙인 CPO

명령문 C 의 의미

연속함수 $\llbracket C \rrbracket \in Memory \rightarrow Memory$ 연속함수 CPO

계산식 E 의 의미

연속함수 $\llbracket E \rrbracket \in Memory \rightarrow \mathbb{Z}_\perp$ 연속함수 CPO

while-문의 의미방정식은

$$\begin{aligned} & \llbracket \text{while } E \ C \rrbracket M \\ &= \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } \llbracket \text{while } E \ C \rrbracket (\llbracket C \rrbracket M) \text{ else } M \end{aligned}$$

다시 쓰면,

$$\begin{aligned} & \llbracket \text{while } E \ C \rrbracket = \\ & \lambda M. \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } \llbracket \text{while } E \ C \rrbracket (\llbracket C \rrbracket M) \text{ else } M. \end{aligned}$$

즉, while-문의 의미 $\llbracket \text{while } E \ C \rrbracket$ 는 연속 함수

$$\begin{aligned} & \lambda X. (\lambda M. \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } X(\llbracket C \rrbracket M) \text{ else } M) \\ & \in (\text{Memory} \rightarrow \text{Memory}) \rightarrow (\text{Memory} \rightarrow \text{Memory}) \end{aligned}$$

의 최소고정점

$$\begin{aligned}
& \llbracket \text{while } E \ C \rrbracket \\
&= \text{lfp } \mathcal{F} \in \text{Memory} \rightarrow \text{Memory} \\
&= \text{lfp}(\lambda X. (\lambda M. \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } X(\llbracket C \rrbracket M) \text{ else } M))
\end{aligned}$$

정의되었는가? $\llbracket \text{while } E \ C \rrbracket$ 는 $\llbracket E \rrbracket$ 와 $\llbracket C \rrbracket$ 를 가지고 조립.

$$M \in \text{Memory} = \text{Var} \rightarrow \text{Value}$$

$$z \in \text{Value} = \mathbb{Z}_{\perp}$$

$$x \in \text{Var} = \text{Program Variable}_{\perp}$$

$$\llbracket C \rrbracket \in \text{Memory} \rightarrow \text{Memory}$$

$$\llbracket E \rrbracket \in \text{Memory} \rightarrow \text{Value}$$

$$\llbracket \text{skip} \rrbracket = \lambda M. M$$

$$\llbracket x := E \rrbracket = \lambda M. M \{x \mapsto \llbracket E \rrbracket M\}$$

$$\llbracket \text{if } E \text{ } C_1 \text{ } C_2 \rrbracket = \lambda M. \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } \llbracket C \rrbracket M \text{ else } \llbracket C_2 \rrbracket M$$

$$\llbracket C_1 ; C_2 \rrbracket = \lambda M. \llbracket C_2 \rrbracket (\llbracket C_1 \rrbracket M)$$

$$\llbracket \text{while } E \text{ } C \rrbracket = \text{lfp}(\lambda X. (\lambda M. \text{if } \llbracket E \rrbracket M \neq 0 \text{ then } X(\llbracket C \rrbracket M) \text{ else } M))$$

$$\llbracket n \rrbracket = \lambda M. n$$

$$\llbracket x \rrbracket = \lambda M. M(x)$$

$$\llbracket E_1 + E_2 \rrbracket = \lambda M. (\llbracket E_1 \rrbracket M) + (\llbracket E_2 \rrbracket M)$$

$$\llbracket - E \rrbracket M = \lambda M. - (\llbracket E \rrbracket M)$$

프로그램 C 의 의미: 조립식

- ▶ C 는 부품들로 조립됨

$$C = AB$$

- ▶ $\llbracket C \rrbracket$ 도 부품의 의미들로 조립됨

$$\begin{aligned}\llbracket AB \rrbracket &= \dots \llbracket A \rrbracket \dots \llbracket B \rrbracket \dots \\ &= \dots 1 + 2 \dots\end{aligned}$$

프로그램 C 의 의미: 고정점

예: $C = AB$

$$\llbracket AB \rrbracket = \llbracket A \rrbracket + \llbracket B \rrbracket$$

여기서

$$\llbracket A \rrbracket = 1$$

$$\llbracket B \rrbracket = \dots \llbracket B \rrbracket \dots$$

즉,

$$\begin{aligned} (\llbracket A \rrbracket, \llbracket B \rrbracket) &= (1, \dots \llbracket B \rrbracket \dots) \\ &= \mathcal{F}(\llbracket A \rrbracket, \llbracket B \rrbracket) \end{aligned}$$

프로그램 C 의 의미: 고정점

$$\llbracket C \rrbracket = \mathcal{F}(\llbracket C \rrbracket)$$

C 의 의미 방정식

$$\llbracket C \rrbracket \in D$$

CPO

$$\mathcal{F} \in D \rightarrow D$$

연속함수

일 때,

$$\llbracket C \rrbracket = \text{lfp } \mathcal{F}$$

C 의 의미

$$= \bigsqcup_{i \in \mathbb{N}} (\mathcal{F}^i \perp)$$

$$= \bigcap \{x \in D \mid \mathcal{F}(x) \sqsubseteq x\}$$