

# 재귀함수 합성을 통해 프로그램 동등성 증명하기

고려대학교 소프트웨어분석 연구실

조민규 이석현 오학주

# 프로그래밍 수업 과제 채점

- 답안 프로그램과 학생 프로그램의 일치 여부로 판단

```
type nat =  
  | Z  
  | S of nat
```

자연수 정의

```
let rec add_sol n1 n2 =  
  match n1 with  
  | Z -> n2  
  | S n -> S (add_sol n n2)
```

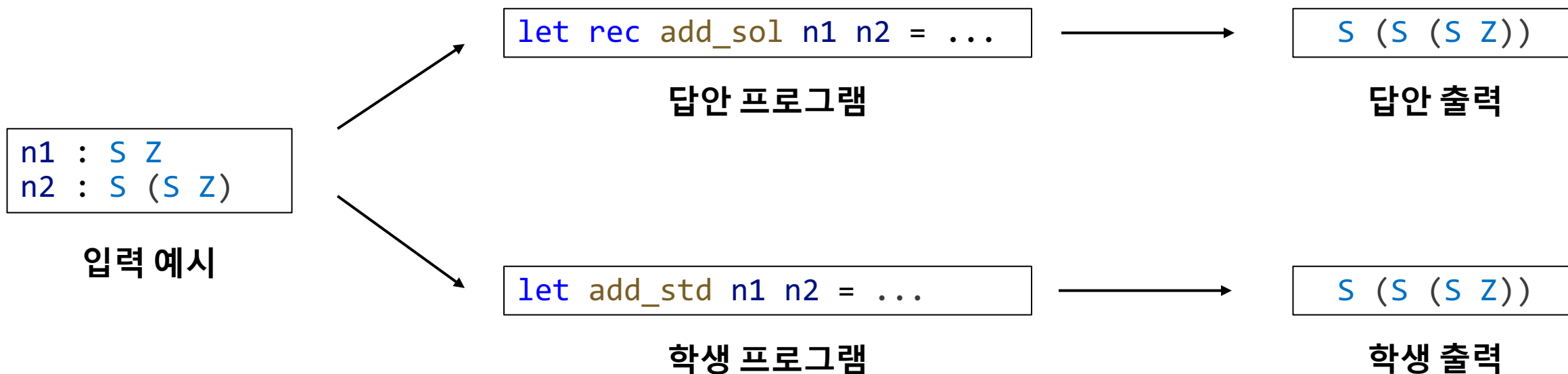
답안 프로그램

```
let rec n2i n =  
  match n with  
  | Z -> 0  
  | S n' -> 1 + n2i n'  
  
let rec i2n i = if i > 0 then S (i2n (i - 1))  
else if i = 0 then Z else failwith "Error"  
  
let add_std n1 n2 = i2n (n2i n1 + n2i n2)
```

학생 프로그램

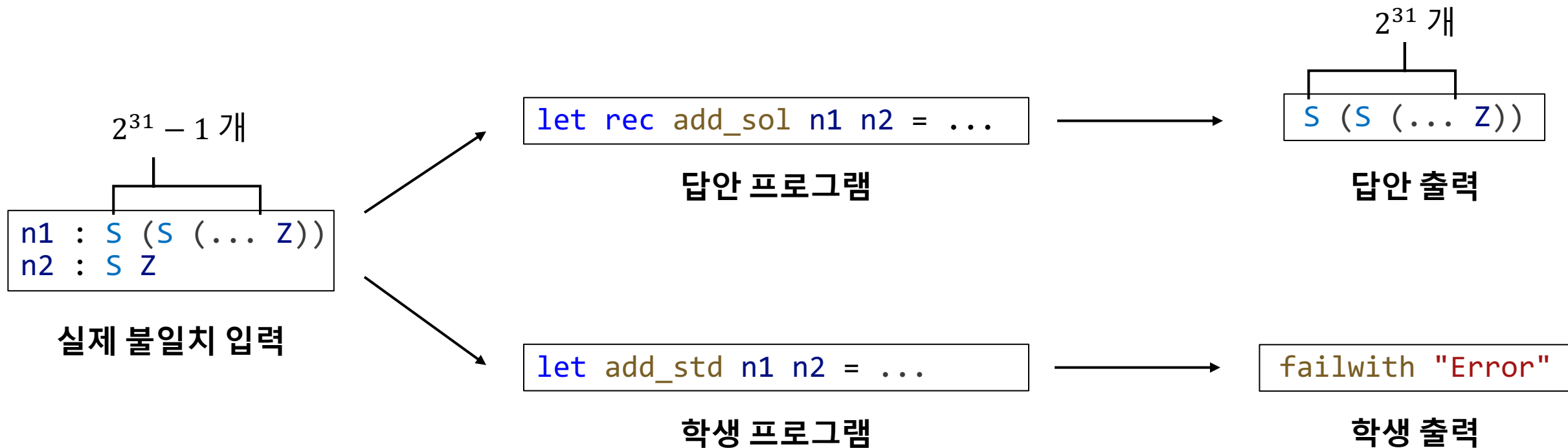
# 프로그래밍 수업 과제 채점

- 테스트 기법이 주로 사용
  - 여러 입력에 따른 두 프로그램의 출력을 비교



# 테스팅 기법의 한계

- 정말 두 프로그램이 일치하는지 **보장하지 않음**



# 동등성 증명 기반 채점

- 두 프로그램의 동등성 증명을 합성

- 아래와 같은 논리식을 자동으로 증명

`forall input, pgm_sol input` **=** `pgm_std input`

- 여러 검증 분야의 핵심 기술로 확장가능

- 최적화 과정 검증
- 컴파일 과정 검증

# 동등성 증명기

- 목표 논리식의 좌우변이 일치할 때까지 가능한 모든 정의/정리를 적용

forall input, `pgm_sol input` = `pgm_std input`

목표 논리식

`let pgm_sol input = ...`

정의1

`let pgm_std input = ...`

정의2



증명 탐색



증명 성공

# 막다른 지점

- 모든 정의/정리를 적용해도 좌우변이 **일치하지 않을 때**

`forall input, pgm_sol input = pgm_std input`

목표 논리식

`let pgm_sol input = ...`

정의1

`let pgm_std input = ...`

정의2



증명 탐색



증명 실패

# 보조정리 탐색

- 좌우변을 일치하게 만드는 보조정리 탐색

`forall input, pgm_sol input = pgm_std input`

목표 논리식

`let pgm_sol input = ...`

정의1

`let pgm_std input = ...`

정의2



증명 탐색



보조정리 탐색



증명 성공

`forall ...,`

`sol_aux ...`

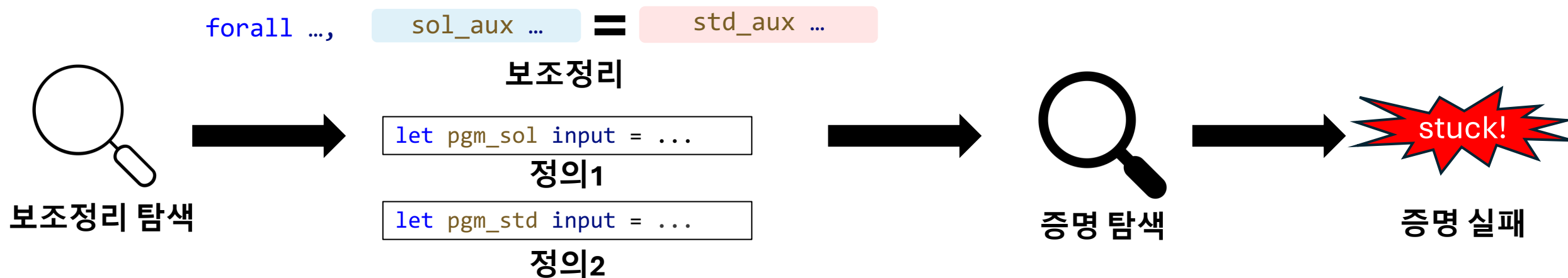
`=`

`std_aux ...`

보조정리

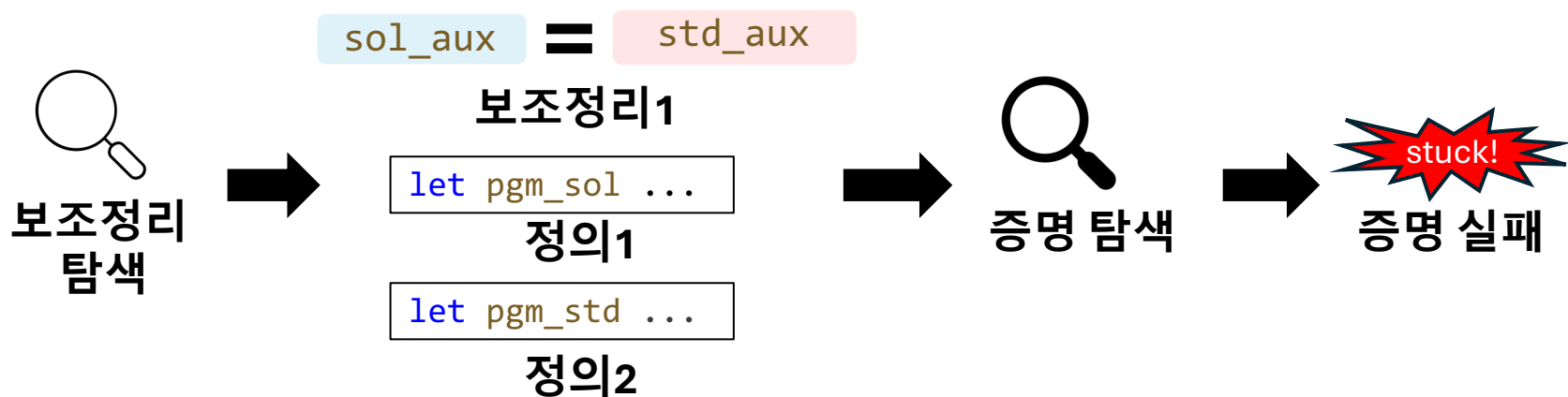
# 기존기술의 한계

- 찾은 보조정리의 **증명이 불가능**한 경우



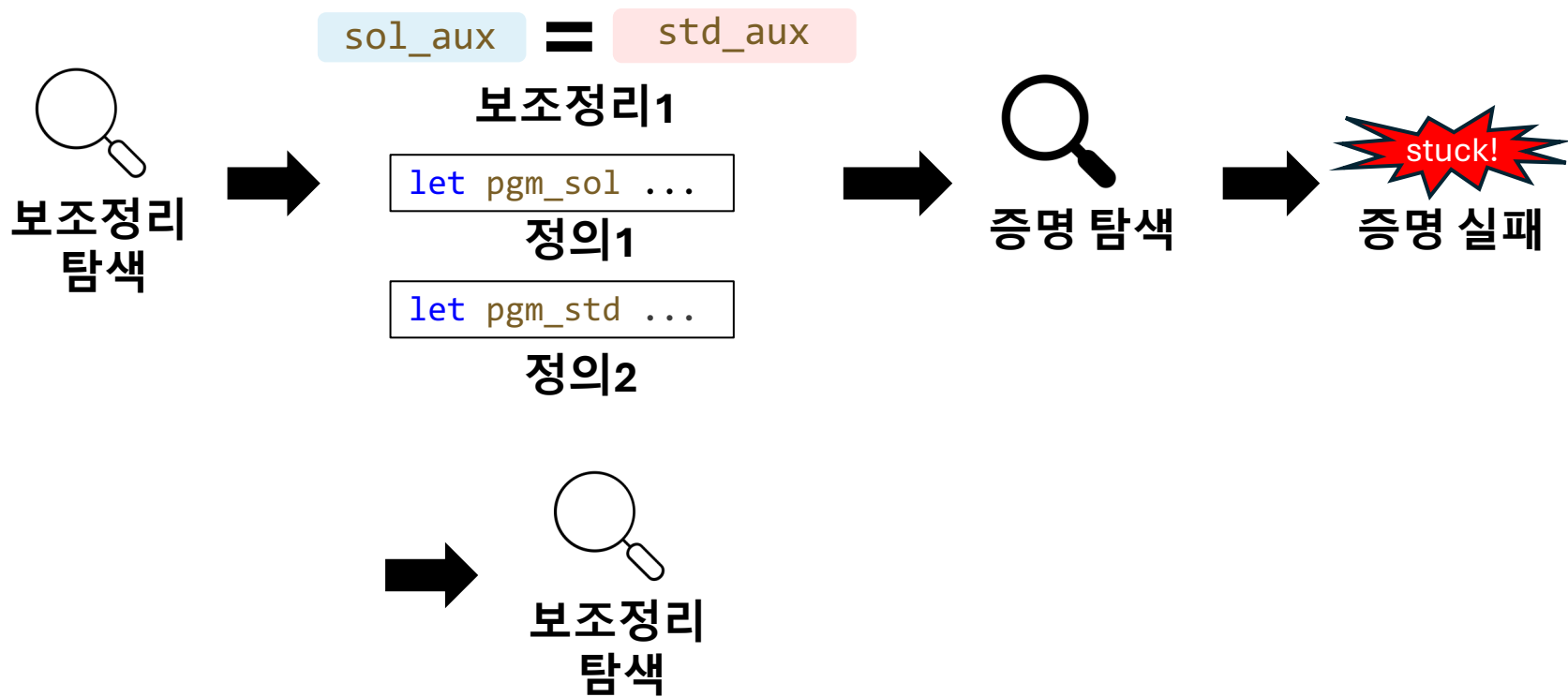
# 기존기술의 한계

- 보조정리 탐색과 증명 실패가 번갈아 계속해서 나타남



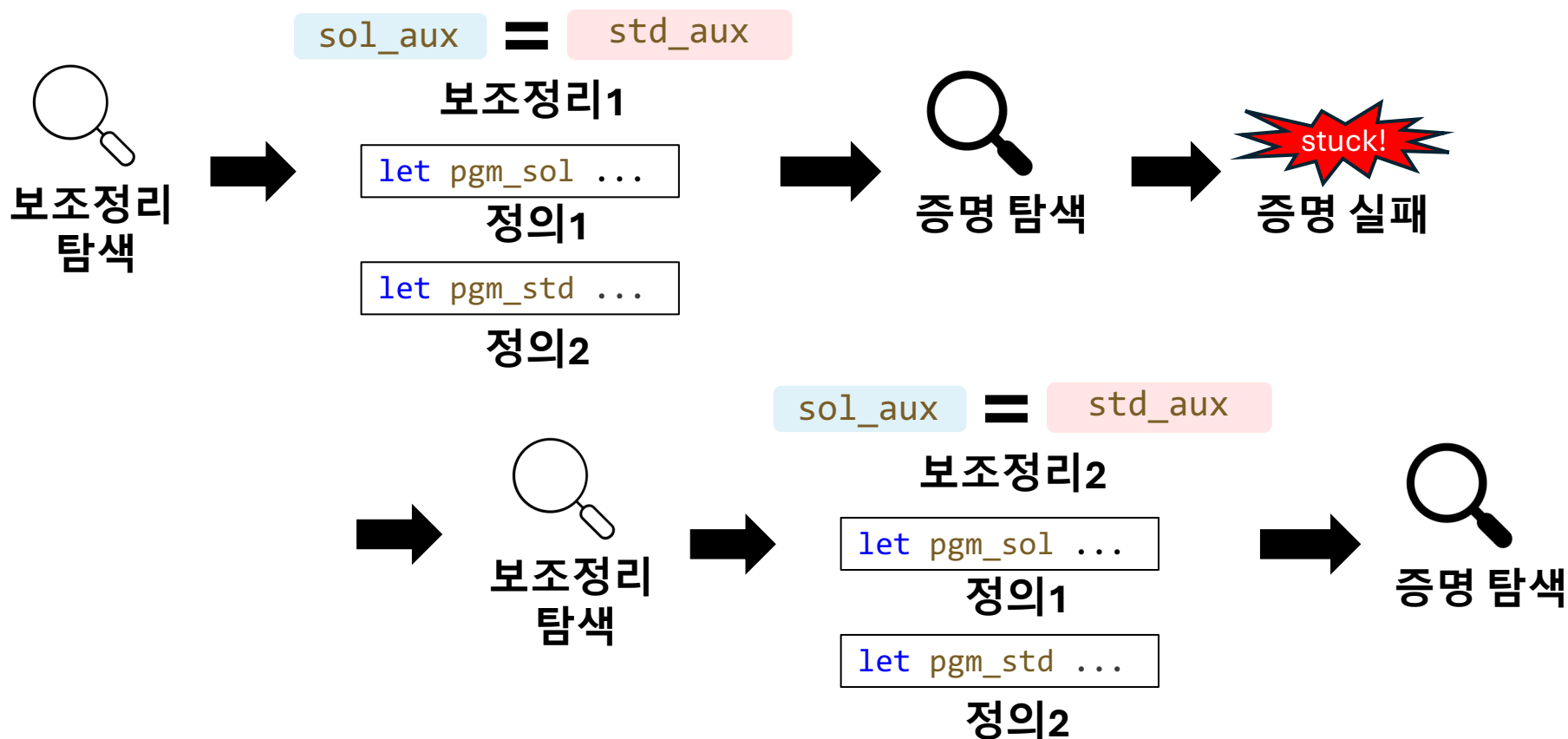
# 기존기술의 한계

- 보조정리 탐색과 증명 실패가 번갈아 계속해서 나타남



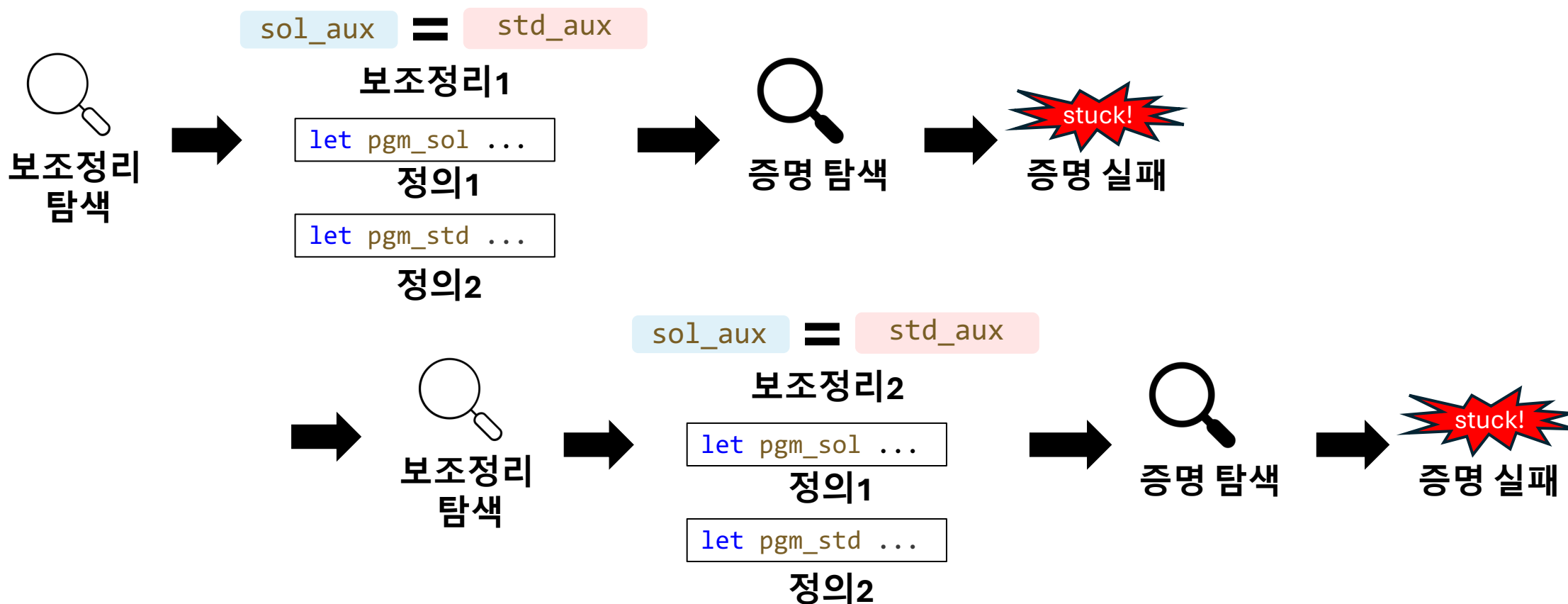
# 기존기술의 한계

- 보조정리 탐색과 증명 실패가 번갈아 계속해서 나타남



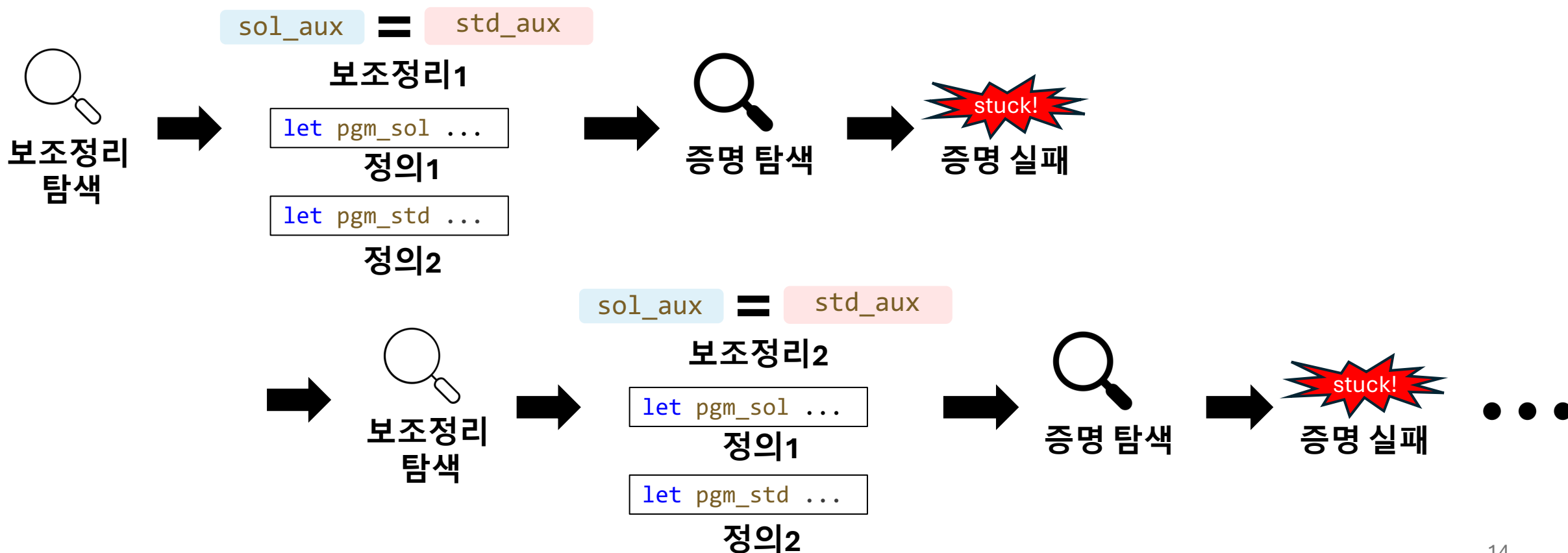
# 기존기술의 한계

- 보조정리 탐색과 증명 실패가 번갈아 계속해서 나타남



# 기존기술의 한계

- 보조정리 탐색과 증명 실패가 번갈아 계속해서 나타남



# 목표

- 찾은 보조정리의 증명이 불가능한 경우



for


미래를 보고  
증명가능한 보조정리를 찾자!



# 포스터에서

## • 동등성 증명 예시와 함께하는

- 기존 기술 및 한계
- 구체적인 아이디어
- 구현 디테일
- 예상 결과



**KOREA UNIVERSITY**  
1909

## 재귀함수 합성을 통한 프로그램 동등성 증명

고려대학교 소프트웨어분석 연구실 조민규 이석현 오학주

### 1. 문제

- 동등성 증명
  - 두 프로그램의 의미가 같음을 증명
  - 다양한 분야에서 신뢰성 확보의 핵심 기술
  - ex) 컴파일러 검증, 최적화 검증, 과제 채점 등
- 동등성 증명 예시: 프로그래밍 과제 채점 상황
  - "주어진 말다 계산식이 달여있는지 검사하는 함수를 구현하시오"

**답안 코드**

```

type lambda =
| V of string
| C of lambda * lambda
| P of string * lambda

let rec f_sol lam vars =
match lam with
| V x -> (* f_sol V case *)
| C (e1,e2) -> (* f_sol C case *)
| P (x,e) -> f_sol e (x::vars)
let main1 lam = f_sol lam []
                    
```

**학생 코드**

```

type lambda =
| V of string
| C of lambda * lambda
| P of string * lambda

let rec f_std lam vars =
match lam with
| V x -> (* f_std V case *)
| C (e1,e2) -> (* f_std C case *)
| P (x,e) -> f_std e (P (x, vars))
let main2 lam = f_std lam (V " ")
                    
```

아래 논리식에서 시작:

$$\text{forall lam, main1 lam} = \text{main2 lam}$$

구조적 귀납법에 의해 세 하위 증명으로 분리

- $(*) f\_sol V \text{ case } *) = (* f\_std V \text{ case } *)$  -- 간단하게 풀림
- $(*) f\_sol C \text{ case } *) = (* f\_std C \text{ case } *)$
- 귀납가정:  $f\_sol e [] = f\_std e (V " ")$

함수정의로 재작성

$$f\_sol e [x] = f\_std e (P (x, e)) (V " ")$$

$$f\_sol e [x] = f\_std e (P (x, V " "))$$

- 막다른 지점
  - 더 이상 사용할 수 있는 함수정의/귀납가정/보조정리가 없는 상태

**목표 : 막다른 지점을 해결하는 보조 정리 찾기**

- 만약 보조 정리가 있다면:
 
$$f\_sol e [x] = f\_std e (P (x, V " "))$$

$$f\_std e (P (x, V " ")) = f\_std e (P (x, V " "))$$

### 2. 기존 기술 및 한계

- CCLemma(ICFP'24)
  - 공통된 표현식을 단일변수로 일반화해 보조정리 수립
  - 한계: 일반화-막다른 지점 고리가 계속해서 나타나면서 증명에 실패하는 경우가 다수

일반화

$$f\_sol e [x] = f\_std e (P (x, V " "))$$

lemmma

$$\text{forall lam x, f\_sol lam [x] = f\_std lam (P (x, V " "))}$$

1,2  $(*) f\_sol V, C \text{ case } *) = (* f\_std V, C \text{ case } *)$

3.  $f\_sol (P (x1, e)) [x] = f\_std (P (x1, e)) (P (x, V " "))$

일반화

$$f\_sol e [x1;x] = f\_std e (P (x1, P (x, V " ")))$$

lemmma

$$\text{forall lam x1 x, f\_sol lam [x1;x] = f\_std lam (P (x1, P (x, V " ")))}$$

3.  $f\_sol (P (x2, e)) [x1;x] = f\_std (P (x2, e)) (P (x1, P (x, V " ")))$

일반화

$$f\_sol e [x2;x1;x] = f\_std e (P (x2, P (x1, P (x, V " "))))$$

### 3. 핵심 아이디어

- 매 막다른 지점마다 규칙적으로 늘어나는 표현식을 포착해 미리 일반화

$$\text{forall e ... xn ... x1 x, } f\_sol e [x1;x] = f\_std e (P (... (P (xn, P (x1, P (x, V " "))))))$$

$$[x1;x1;x] \quad P (... (P (xn, P (x1, P (x, V " "))))$$

lemmma

$$\text{forall lam lst, f\_sol lam lst = f\_std lam (lst2lambda lst)}$$

미리 일반화하였기 때문에 귀납가정을 이용해 증명 가능:

귀납가정:  $\text{forall lst, f\_sol e lst} = f\_std e (lst2lambda lst)$

$$f\_sol (P (x,e)) lst = f\_std (P (x, e)) (lst2lambda lst)$$

$$f\_sol e x::lst = f\_std e (lst2lambda x::lst)$$

귀납가정으로 재작성

$$f\_sol e x::lst = f\_sol e x::lst \quad \checkmark$$

### 4. 전체 구조

프로그램/  
동식

증명 탐색

**Dilemma**

막다른 지점

규칙 탐지

함수 합성

단순 일반화

증명 완료

재귀함수/  
보조정리

### 5. 함수 합성 방법

- 반복되는 부분 추출

$$f\_student e (V " ") \quad f\_student e (P (x1, V " ")) \quad f\_student e (P (x2, P (x1, V " ")))$$

$$e1 \quad e2 \quad e3$$

$$x1 \quad x2 \quad x3$$

$$e2 - e1 \quad e3 - e2 \quad \dots$$

$$x1 \quad x2 \quad \dots \quad xn$$

- 이웃한 항의 차를 구하고 공차로 확장

- 첫번째 항과 공차를 이용해 함수로 변환

$$e1 \quad d1$$

$$x1 \quad x2$$

let rec lst2lambda x\_list =  
match x\_list with  
| [] -> V " "  
| x::tl -> P (x, lst2lambda tl)

### 6. 평가

문제집	문제수	CCLemma	Dilemma	
			함수 합성 X	함수 합성 O
IsaPlanner	87	76	76	>76
CLAM	50	37	41	>41
Optimization	96	22	20	>20
TestML	49	14	17	>17
Total	282	148	154	>154