

프로그래밍 과제에서 작은 오류의 자동 판별 시스템

이석현, 전민석, 오학주
소프트웨어 분석 연구실, 고려대학교

2023.02.02.

SW재난연구센터 워크샵 @지지향, 파주



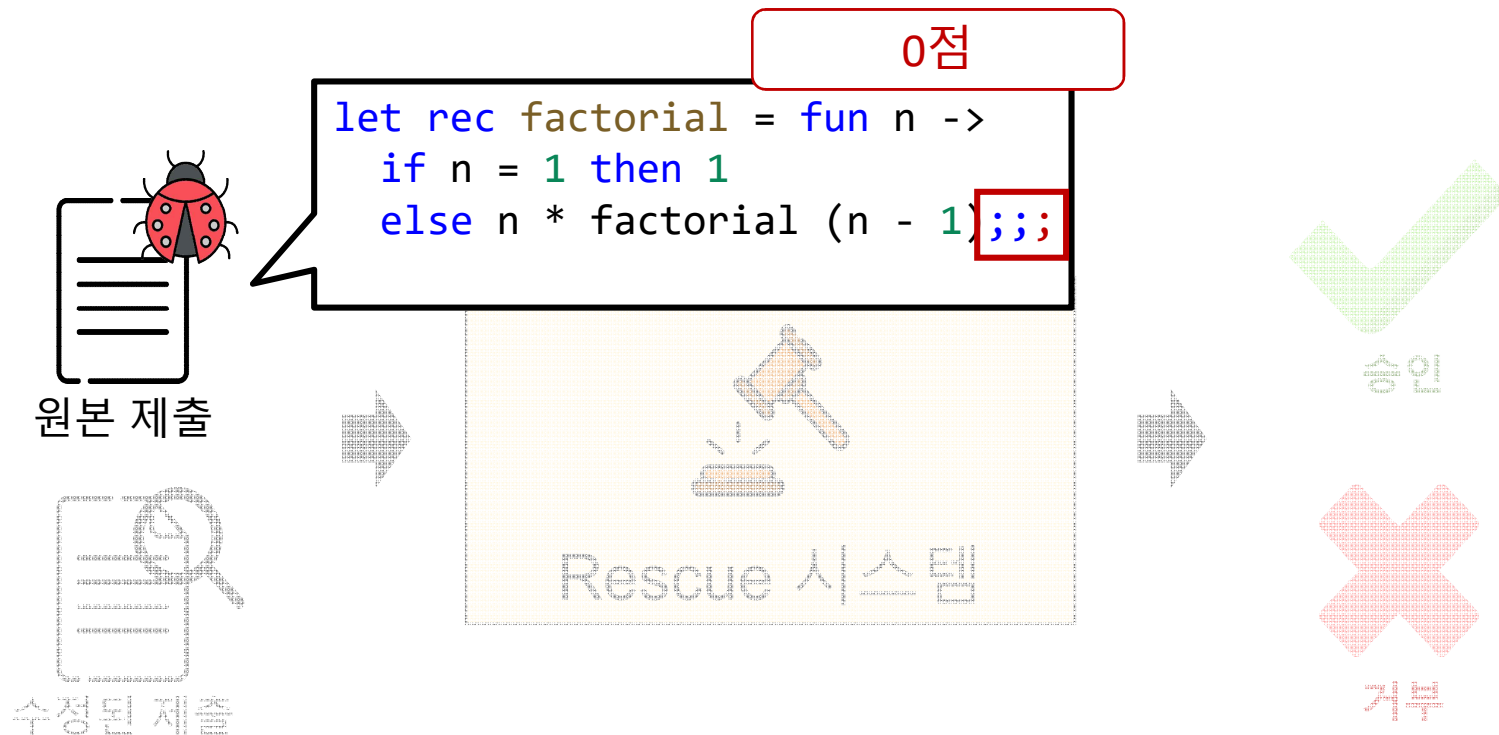
목표: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템



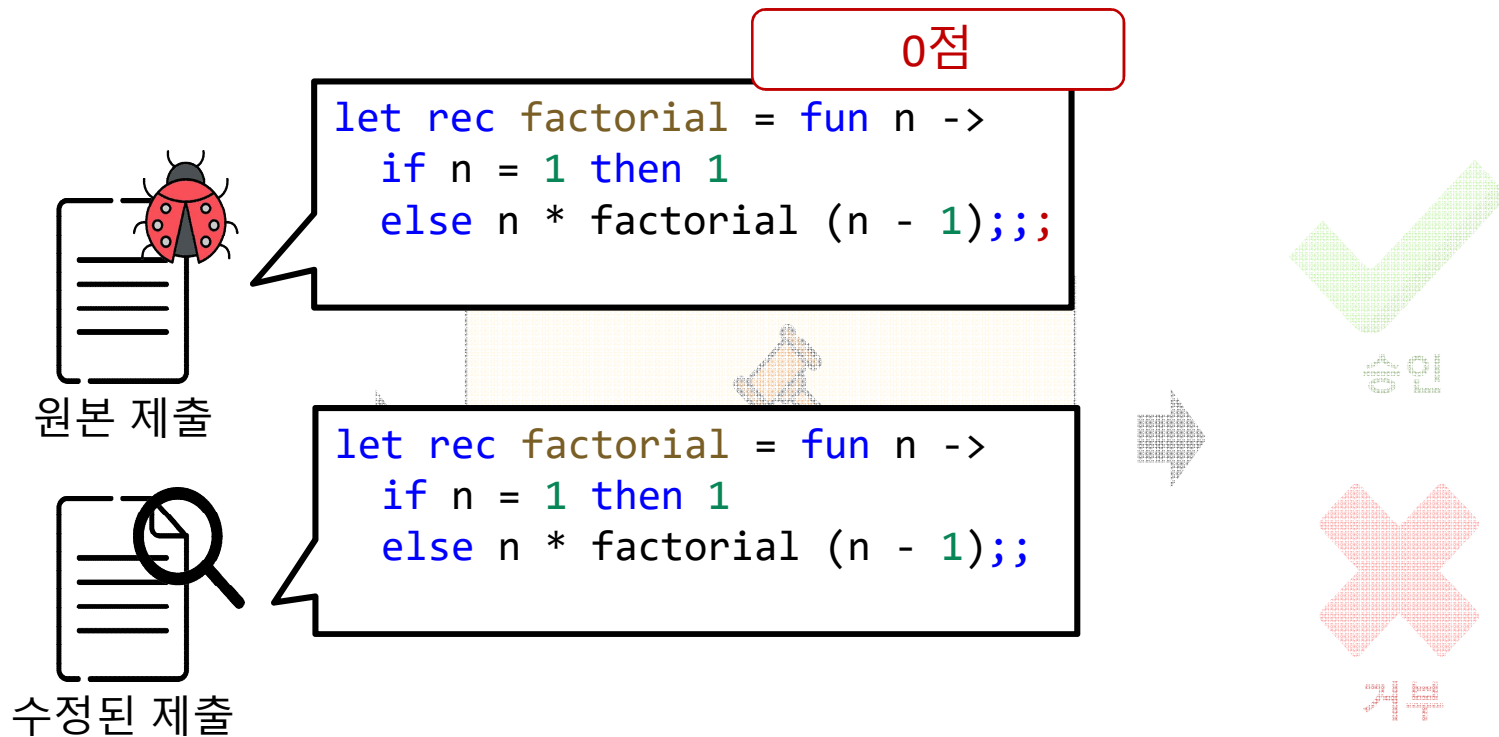
목표: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템



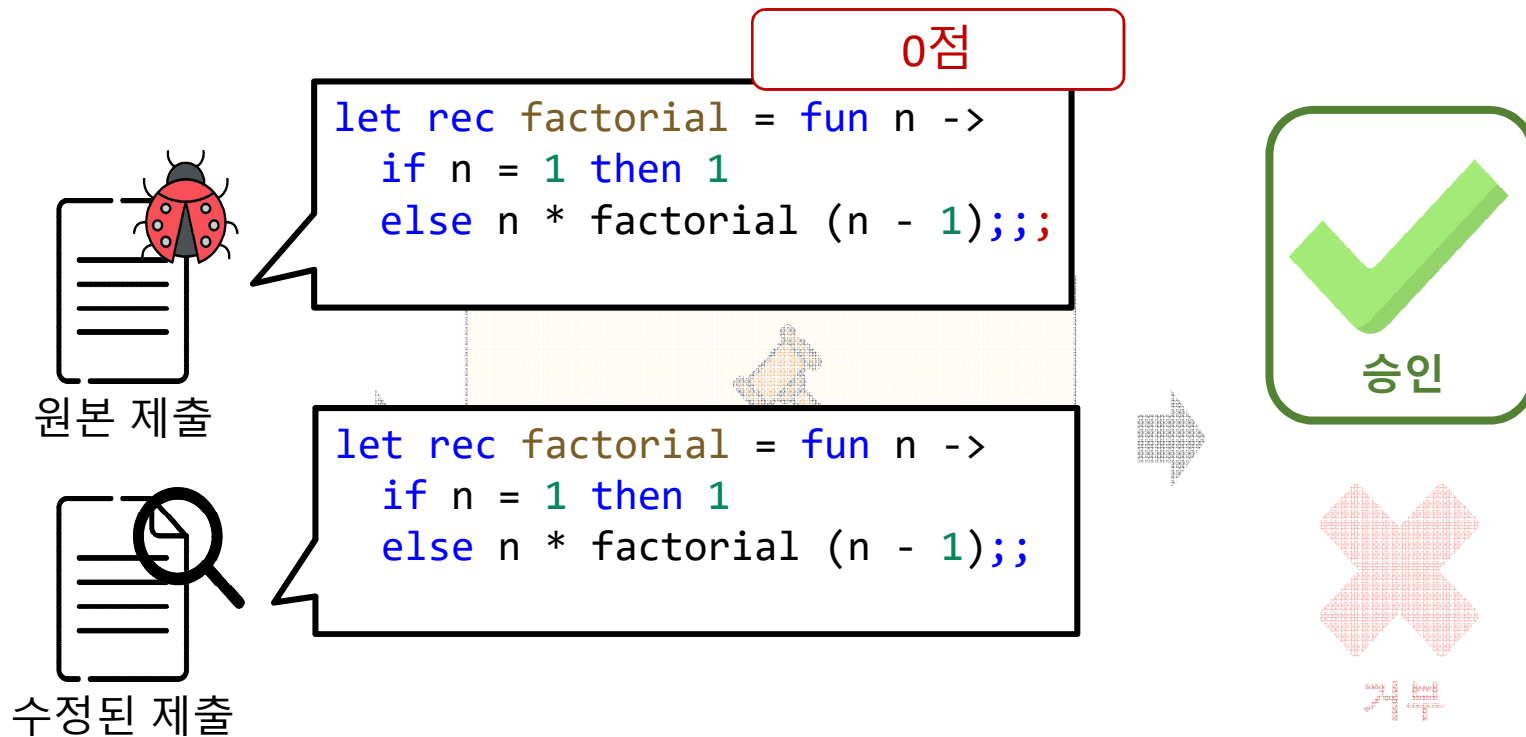
목표: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템



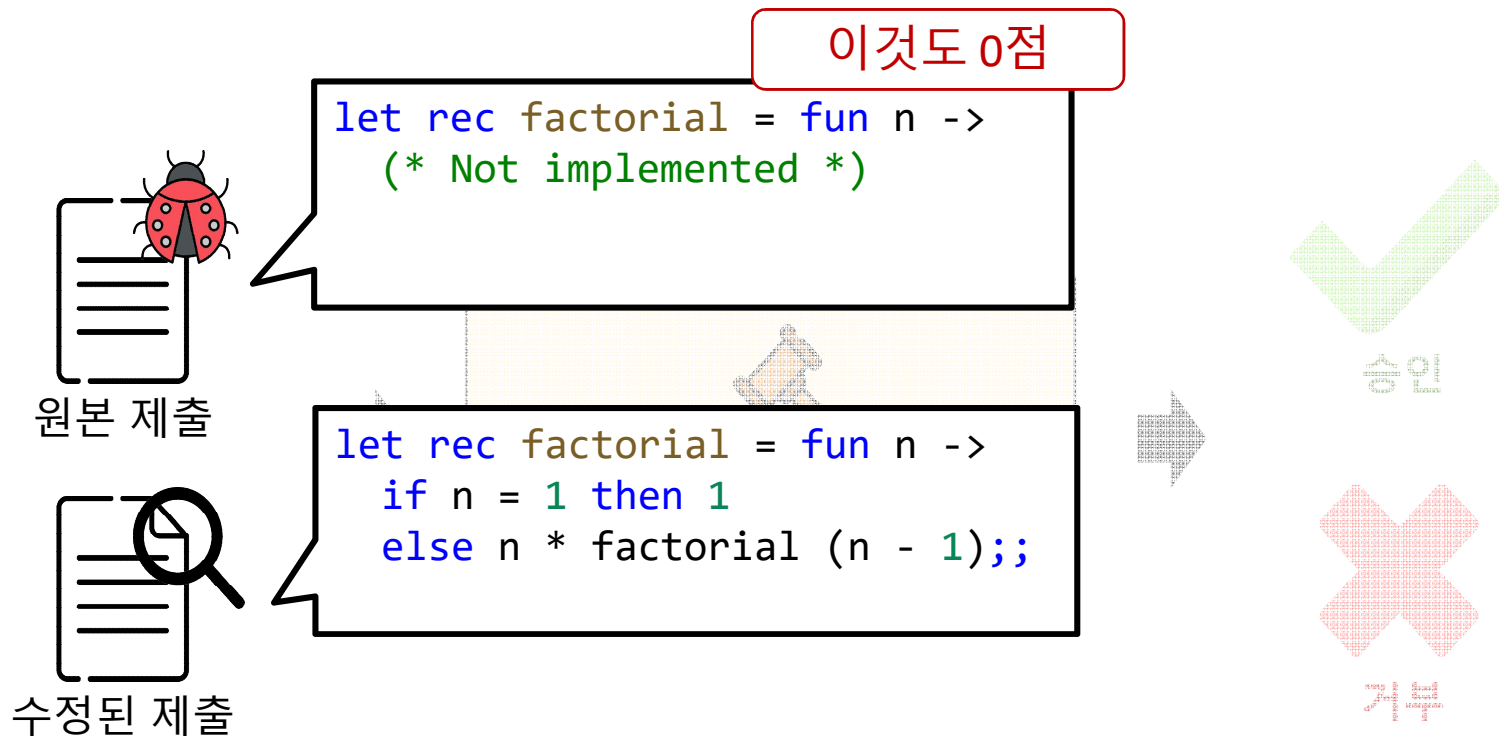
목표: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템



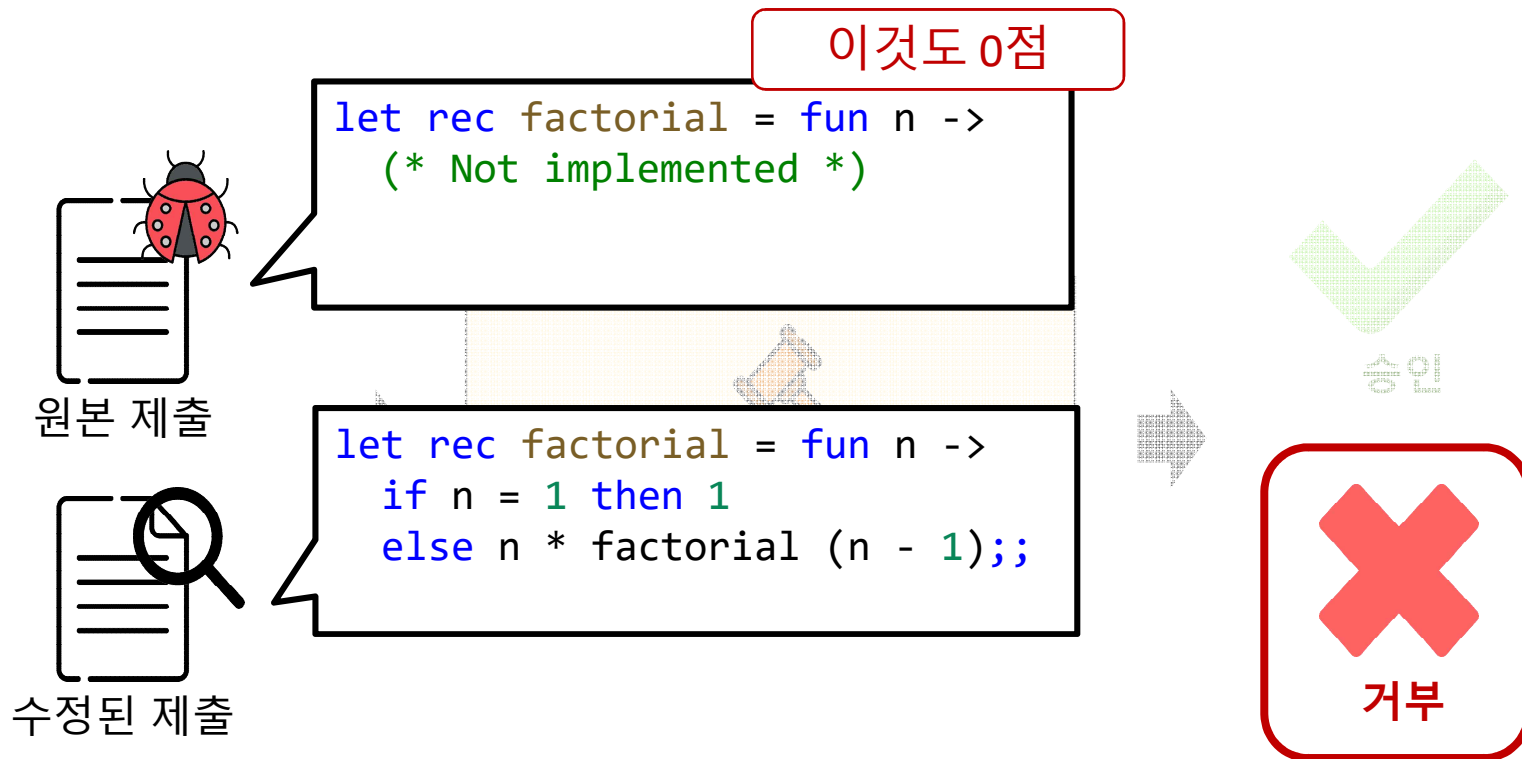
목표: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템




목표: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템



이번 학기 PL 조교를 맡아보자.

 프로그래밍언어 조교

민석, 석현,

이번 학기 프로그래밍언어 조교를 맡아보자.

주로 할 일은

- 숙제 채점 및 피드백
- 강의실 상황 체크 (프로젝터, 마이크)

이번 학기는 ...

이번 학기 PL 조교를 맡아보자.



프로그래밍언어 조교

민석, 석현,

이번 학기 프로그래밍언어
주부 학원

숙제 채점 및 피드백

강의실 상황 제1 (프로그래밍언어)

이번 학기 ...

평가

- 과제 – 90%
 - 5-6개의 프로그래밍 과제
 - 지각 제출은 허용 안함
- 출석 – 10%

조교님, 다름이 아니라 ...

✉ HW1 채점 관련해서 질문드립니다.

0점

안녕하세요 조교님!

다름이 아니라 1번문항에서 조교님께 질문을 드리기 위해서
(* 안녕하세요 조교님! 혹시 실험해본 테스트케이스들을
남겨두고 주석처리 해두어도 될까요?)

이렇게 주석처리를 통해 질문을 남겼으나 저의 불찰로 인해서
주석을 완벽하게 닫지 못했습니다TT.

*)로 주석을 닫지 못하여 ...

조교님, 다름이 아니라 ...



HW1 채점 관해서 질문드립니다.

0점

```
let rec fib : int -> int
= fun n -> match n with
| 0 -> 0
| 1 -> 1
| _ -> fib(n - 1) + fib(n - 2);;
```

(* 안녕하세요 조교님!
혹시 실험해본 테스트케이스들을 남겨두고 주석처리 해두어도 될까요?)

조교님, 다름이 아니라 ...



HW1 채점 관해서 질문드립니다.

0점

```
let rec fib : int -> int
= fun n -> match n with
| 0 -> 0
| 1 -> 1
| _ -> fib(n - 1) + fib(n - 2);;
```

(* 안녕하세요 조교님!

혹시 실험해본 테스트케이스들을 남겨두고 주석처리 해두어도 될까요?)

조교님, 다름이 아니라 ...

✉ HW1 채점 관련해서 질문드립니다.

✉ HW1 1번 채점

0점

제가 이번 HW1에 1번인 fibonacci 문제에서 공지사항에 6번을 제대로 못 읽어서

let rec fib을 모르고 let rec fibonacci로 바꿨습니다..

이러한 실수에 반성하고 있습니다. 죄송합니다.

하지만 코드가 맞고, sample testcase에서 주신대로 다 적었는데 똑같은 답(return result)이 나왔습니다.

제가 열심히 했는데, 혹시 이번만이라도 봐줄수 있는지 여쭙보고 싶습니다.

조교님, 다름이 아니라 ...

✉ HW1 채점 관련해서 질문드립니다.

✉ HW1 1번 채점

✉ 프로그래밍언어 HW5 채점 문의

0점

안녕하세요.

제가 제출한 HW5 코드에서, 마지막 줄에 과제 구현과 무관한
typeof(...) 구문에서 syntax error가 발생하여 모든
테스트케이스가 0점 처리되었습니다. 아무래도 복사/붙여넣기
과정에서 발생한 실수로 보입니다.

이 줄을 지우고 ...

조교님, 다름이 아니라 ...

✉ HW1 채점 관련해서 질문드립니다.

✉ HW1 1번 채점

✉ 프로그래밍언어 HW5 채점 문의

✉ 프로그래밍 언어 Homework 4 채점 결과에 대해

0점

다름이 아니라, 제가 Homework 4를 완성하고 나서 과제를 제대로 했는지 검사하기 위해 사용했던 테스트 케이스들을 TryML의 블록 주석 기능으로 주석처리한 뒤에 submit을 했습니다.

이 후 다시 수정할 일이 있을지도 몰라 지우지 않고 놔둔 채로 시간이 지나 과제가 제출이 되었는데요, 오늘 과제 점수를 확인해 보니 0점이어서 제출한 과제를 확인해 보니 주석이 끝나지 않아 compile error가 나고 있었습니다.

주석 처리를 하고 ...



HW1 채점 관련해서 질문드립니다.



HW1 1번 채점

조교님, 다름이 아니라 ...



프로그래밍언어 HW5 채점 문의



프로그래밍 언어 Homework 4 채점 결과에 대해 ...



프로그래밍언어 과제3에 대한 문의입니다.

30점 감점

안녕하세요 조교님

과제3에서 error 메시지를 UndefinedSemantics 가 아닌 Error 메시지로 제출했습니다.

제가 학점 교류생이라 누구한테 물어볼 사람 없이 너무 너무 힘들게 열심히 코드를 짰는데 에러 메시지를 다르게 써서 점수가 다 깎인 것 같아 너무 속상합니다.

Error message 잘못 입력한 것은 ...

✉ HW1 채점 관련해서 질문드립니다.

✉ HW1 1번 채점

✉ 프로그래밍언어 HW5 채점 문의

조교님, 다름이 아니라 ...

✉ 프로그래밍 언어 Homework 4 채점 결과에 대해 ...

✉ 프로그래밍언어 과제3에 대한 문의입니다.

✉ [프로그래밍언어] HW5 점수 문의드립니다.

0점

안녕하세요.

HW5 정상 제출하였지만
0점이 나와서 다시 확인하여 보니 주석 처리된 부분에서
오류가 발생한 것 같습니다.

코드는 그대로 있는 상태에서
주석을 모두 지웠을 때에는 정상 동작하는데 ...

HW1 채점 관련해서 질문드립니다.

HW1 1번 채점

프로그래밍언어 HW5 채점 문의

프로그래밍 언

프로그래밍언

[프로그래밍언

조교님, 다름이 아니라 ...

안녕하세요.

HW5 정상 제출하
0점이 나와서 다
오류가 발생한 것

코드는 그대로 있
주석을 모두 지웠

- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- 이석현[대학원석.박사통합과정재학 / 컴퓨터학과]
[받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- 이석현[대학원석.박사통합과정재학 / 컴퓨터학과]
[받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- 이석현[대학원석.박사통합과정재학 / 컴퓨터학과]
[받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- 이석현[대학원석.박사통합과정재학 / 컴퓨터학과]
[받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- 이석현[대학원석.박사통합과정재학 / 컴퓨터학과]
[받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- 이석현[대학원석.박사통합과정재학 / 컴퓨터학과]
[받은메일함] Re: 프로그래밍언어 HW5 채점 문의
- [받은메일함] Re: 프로그래밍언어 HW5 채점 문의

하 ...

0점

본에서

학생들이 원하는 것

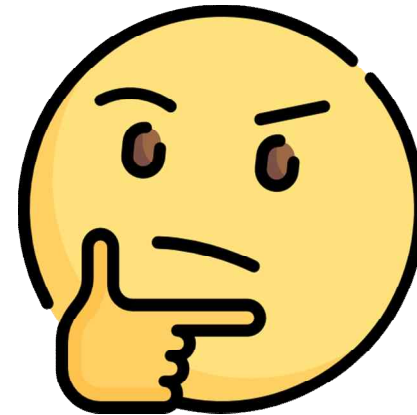
실수로 ...

... 봐주실 수 없나요?

... 재채점은 힘들까요?

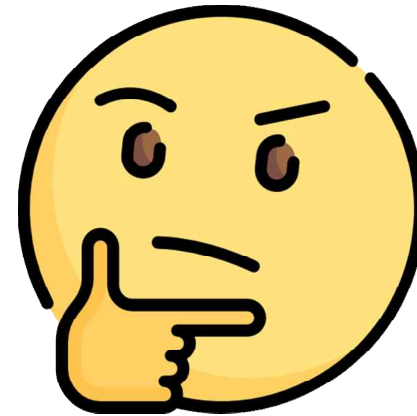
... 다시 확인해 주실 수 있을까요?

... 구제 해주실 수 있을까요?



학생들이 원하는 것

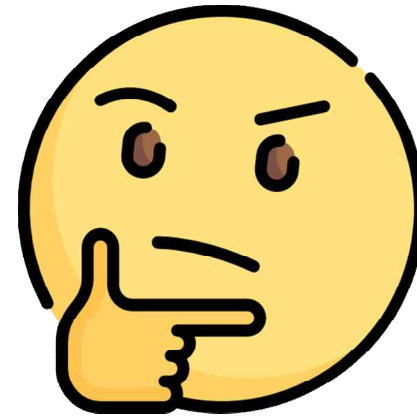
코드에 작은 실수가 있는데
고칠 수 있는 기회를 주세요



학생들이 원하는 것

1. 어떻게 작은 실수를 정의할지?

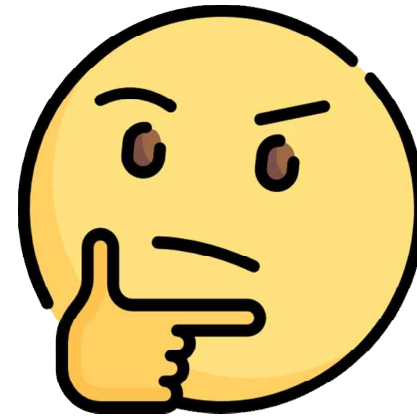
코드에 **작은 실수**가 있는데
고칠 수 있는 기회를 주세요



학생들이 원하는 것

1. 어떻게 작은 실수를 정의할지?

코드에 작은 실수가 있는데
고칠 수 있는 기회를 주세요



2. 어떤 방식으로 기회를 줄지?

초기 수정 언어

- 사소한 수정을 정의하는 "수정 언어"

$$Revise : Code \times Cursor \rightarrow Code \times Cursor$$

$$Code = (Char^*)^*$$

$$Revise = Cmd^*$$


$$Cmd = Move \mid Backspace \mid Insert(Str)$$

$$Move = \circ \mid \leftarrow \mid \rightarrow \mid \uparrow \mid \downarrow$$

$$Str = _ \mid " \mid (\mid) \mid ErrorMessage$$

초기 수정 시스템

- 사소한 수정을 정의하는 "수정 언어"

 프로그래밍언어 HW1 수정 언어

안녕하세요.

HW1에 1번에 아래 수정 언어를 적용하고 싶습니다.

[illegible]

수정 언어는 아래 코드의 마지막 세미콜론을 지웁니다.

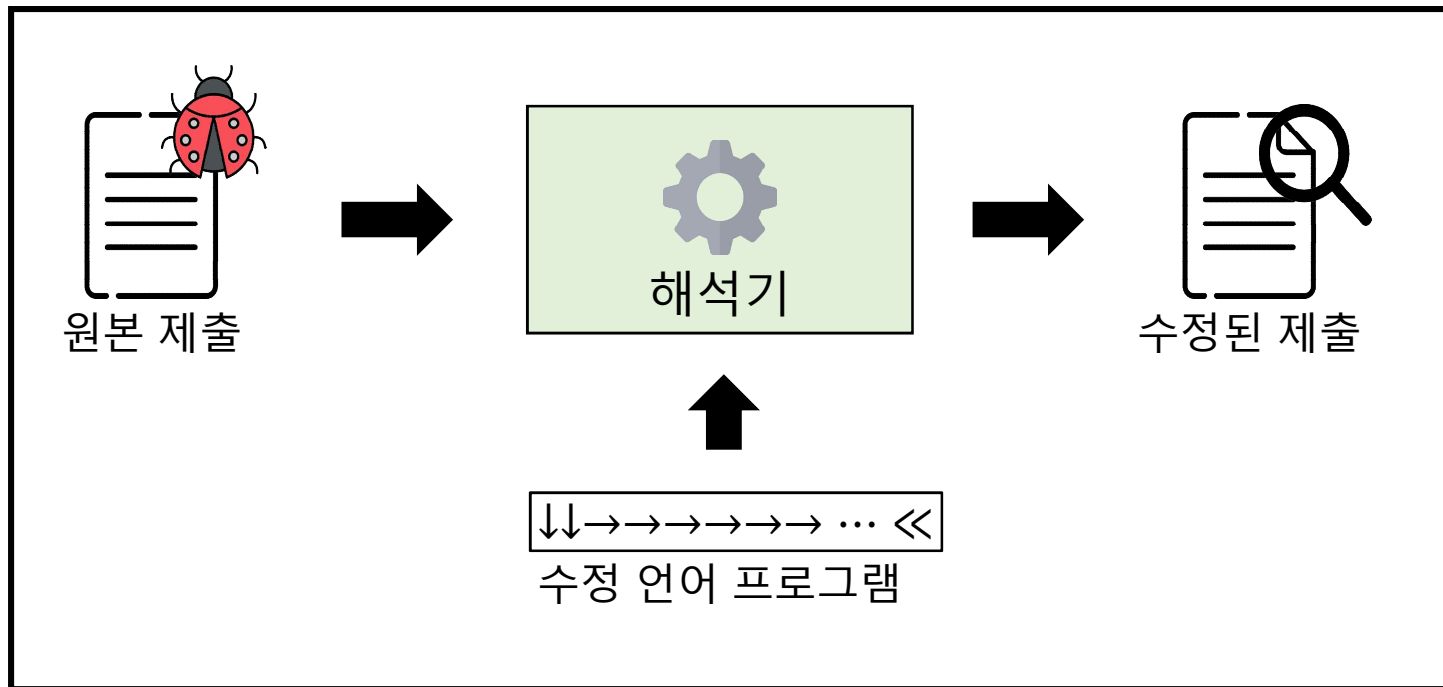
```
1| let rec factorial = fun n ->
2|   if n = 1 then 1
3|   else n * factorial (n - 1);;
```

감사합니다.

초기 수정 시스템

- 사소한 수정을 정의하는 “수정 언어”

✉ 프로그래밍언어 HW1 수정 언어



초기 수정 시스템

- 총 119명의 수강생 중 약 40명의 사소한 수정

안녕하십니까 조교님.

프로그래밍 언어를 수강하고 있는

라고 합니다.

안녕하세요 조교님,

교수님의 프로그래밍언어 강의를 듣고 있는

이라고합니다.

다름이 아니라 hw
립니다.

조교님, 안녕하세요? 이번학기 프로그래밍언어 수강중인

입니다.

지금까지 제가 작성한 과제 코드에 대한 Rescue 프로그램 두 가지를 다음과 같이 제출합니다.

고싶어 메일 드립니다.

하면 좋을 것 같아 pdf로 제출

안녕하세요.

222R 프로그래밍언어

새로 제시된 언어로

origin v(2) >(4) backS

origin v(5) >(8) backS

origin v(8) >(21) back

origin v(12) >(8) back

origin v(15) >(16) bac

origin v(15) >(26) bac

HW2 uniq

> 37 backspace 24

맨 첫줄의 함수 타입 선언이 잘못되어 일부 테스트케이스가 통과되지 않는 문제점을 해결하기 위하여 함수의 타입 선언 부분을 삭제하는 프로그램입니다.

HW3 ml_minus

v 108 > 37 insert(raise UndefinedSemantics)

109번 라인에서 0으로 나누는 경우 발생하는 예외에 대해서 제가 자체적으로 작성한 exception을 발생 시켜 일부 테스트케이스가 통과되지 않는 문제점을 해결하기 위하여 해당 exception을 삭제하고, UndefinedSemantics를 삽입하는 프로그램입니다.

insert()> 7backspaceinsert(
einsert(____)> 8backspaceinsert

36insert(raise

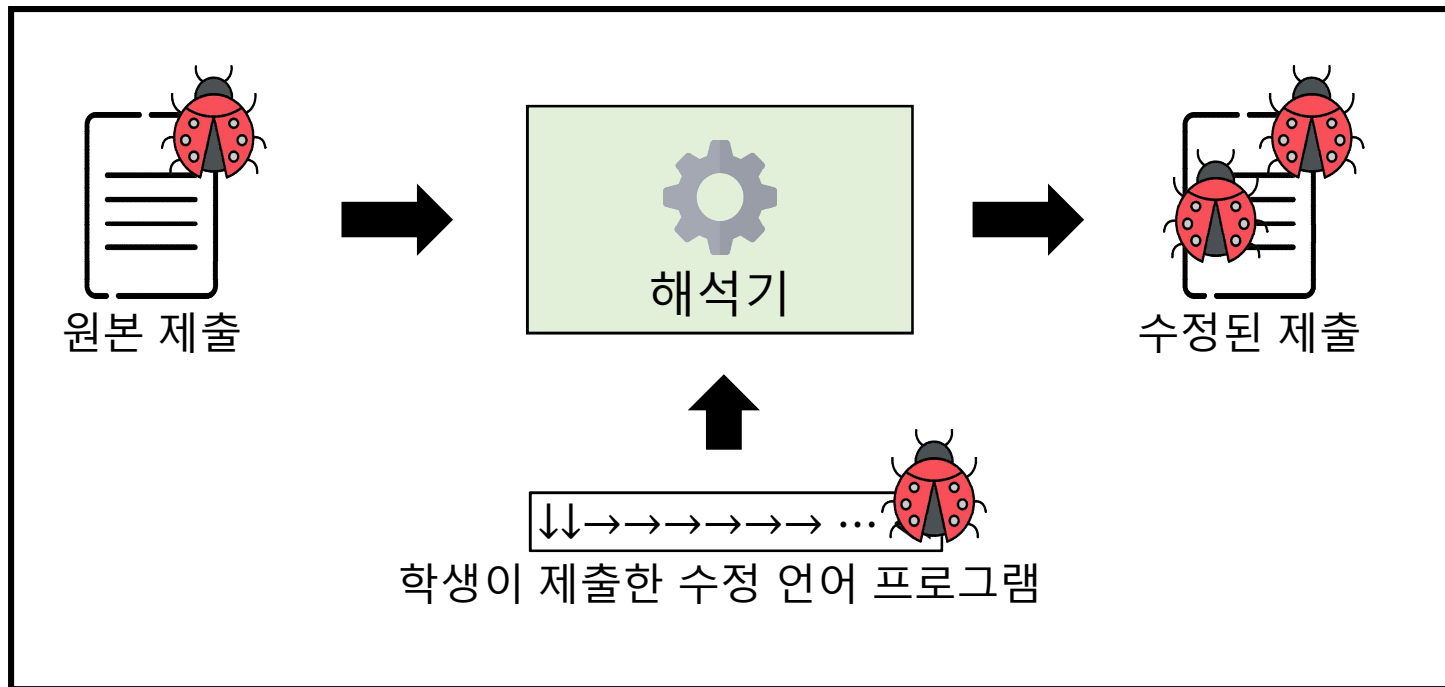
suml로 잘못 표시된

ssuumml로 표시된 문제에서 필요한 진입 함수는 __entry__로 고쳤습니다.

> v6<<<<<<backspace36Insert(raise

초기 수정 시스템의 문제점

- 약 80%의 틀린 수정 언어 프로그램



초기 수정 시스템의 문제점

- 약 80%의 틀린 수정 언어 프로그램

안녕하세요.
프로그래밍언어 조교 이석현입니다.

보내주신 구제 프로그램을 아래와 같이 수정
origin v342 >16 backSpace4 insert(

안녕하세요.
프로그래밍언어 조교 이석현입니다.

구제 프로그램을 작성하느
HW4에 해당하는 구제 프
v227backspace86v8>3

안녕하세요.
프로그래밍언어 조교 이석현입니다.

보내주신 구제 프로그램을 아래와 같이 수정하여 사용하였습니다.

origin v84 >51 backSpace36 insert(raise UndefinedSemantics)
origin v92 >51 backSpace36 insert(raise UndefinedSemantics)
origin v100 >51 backSpace36 insert(raise UndefinedSemantics)
origin v108 >51 backSpace36 insert(raise UndefinedSemantics)
origin v117 >51 backSpace36 insert(raise UndefinedSemantics)
origin v125 >51 backSpace36 insert(raise UndefinedSemantics)
origin v133 >51 backSpace36 insert(raise UndefinedSemantics)
origin v142 >51 backSpace36 insert(raise UndefinedSemantics)

안녕하세요.
프로그래밍언어 조교 이석현입니다.

보내주신 구제 프로그램을 아래와 같이 수정하여 사용하였습니다

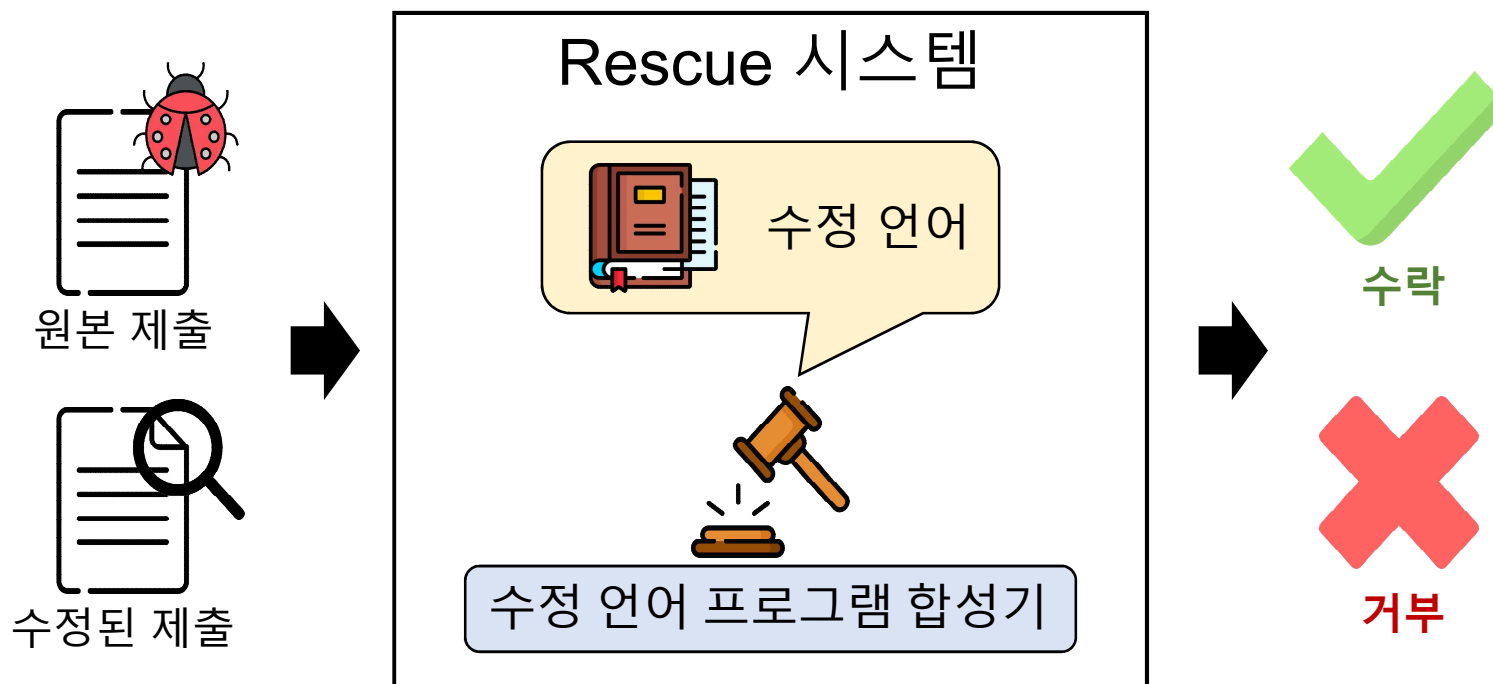
origin v2 >4 >4 backSpace4 insert(_entry_)
origin v5 >8 >4 backSpace4 insert(____)
origin v8 >21 >4 backSpace4 insert(____)
origin v12 >8 >4 backSpace4 insert(____)
origin v15 >16 >4 backSpace4 insert(____)
origin v15 >26 >4 backSpace4 insert(____)

안녕하세요.
프로그래밍언어 조교 이석현입니다.

구제 프로그램을 작성하느라 수고하셨습니다.
세번째 과제에 대한 구제 프로그램은 아래와 같이 수정하여 사용하였습니다.
v 108 > 37 > 46 backspace 41 insert(raise UndefinedSemantics)

Rescue: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템
 - 수정 언어 프로그램 합성이 성공하면 **승인**
 - 수정 언어 프로그램 합성이 실패하면 **거부**



Rescue 데모 1

○ /workspaces/asmr (main x) |



```
--- ocaml/case1/original.ml    2022-12-21 08:58:17.038058924 +0000
+++ ocaml/case1/revised.ml     2022-12-21 08:58:23.889968332 +0000
@@ -1,4 +1,4 @@
-let rec exist : int -> 'a list -> bool
+let rec exist
= fun n lst ->
  match lst with
  | [] -> false
```

Rescue 데모 2

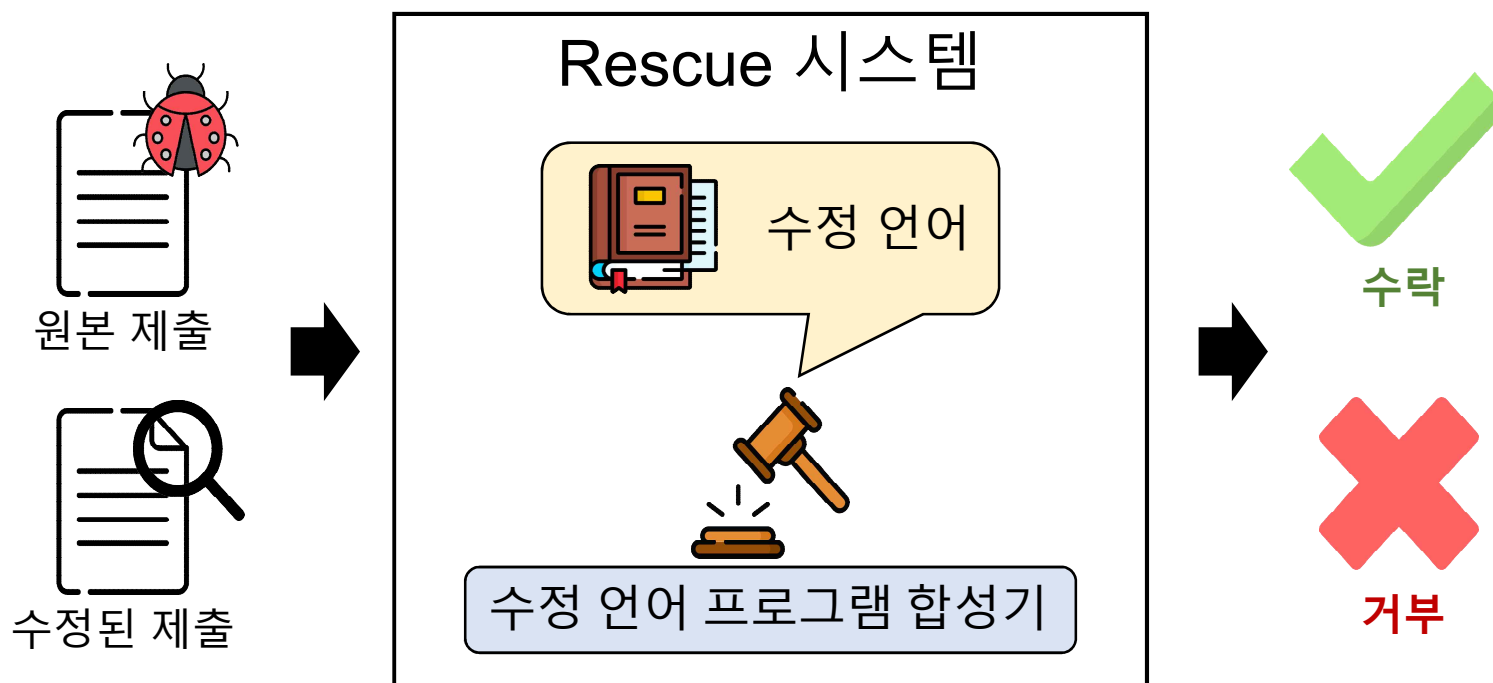
```
○ /workspaces/asmr (main x) |
```

```
--- ocaml/case4/original.ml    2023-01-02 04:31:47.408728495 +0000
+++ ocaml/case4/revised.ml    2023-01-02 04:32:31.196249124 +0000
@@ -306,7 +306,7 @@
     let y_loc = lookup_loc_env y env in
     let f_env2 = extend_env (LocBind (x, y_loc)) f_env in
     callr_extend env f_env2 mem x_t1 y_t1
- | [], [] -> env
+ | [], [] -> f_env
| _ -> raise UndefinedSemantics

let runb : exp -> value
```

Rescue: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템
 - 수정 언어 프로그램 합성이 성공하면 **승인**
 - 수정 언어 프로그램 합성이 실패하면 **거부**



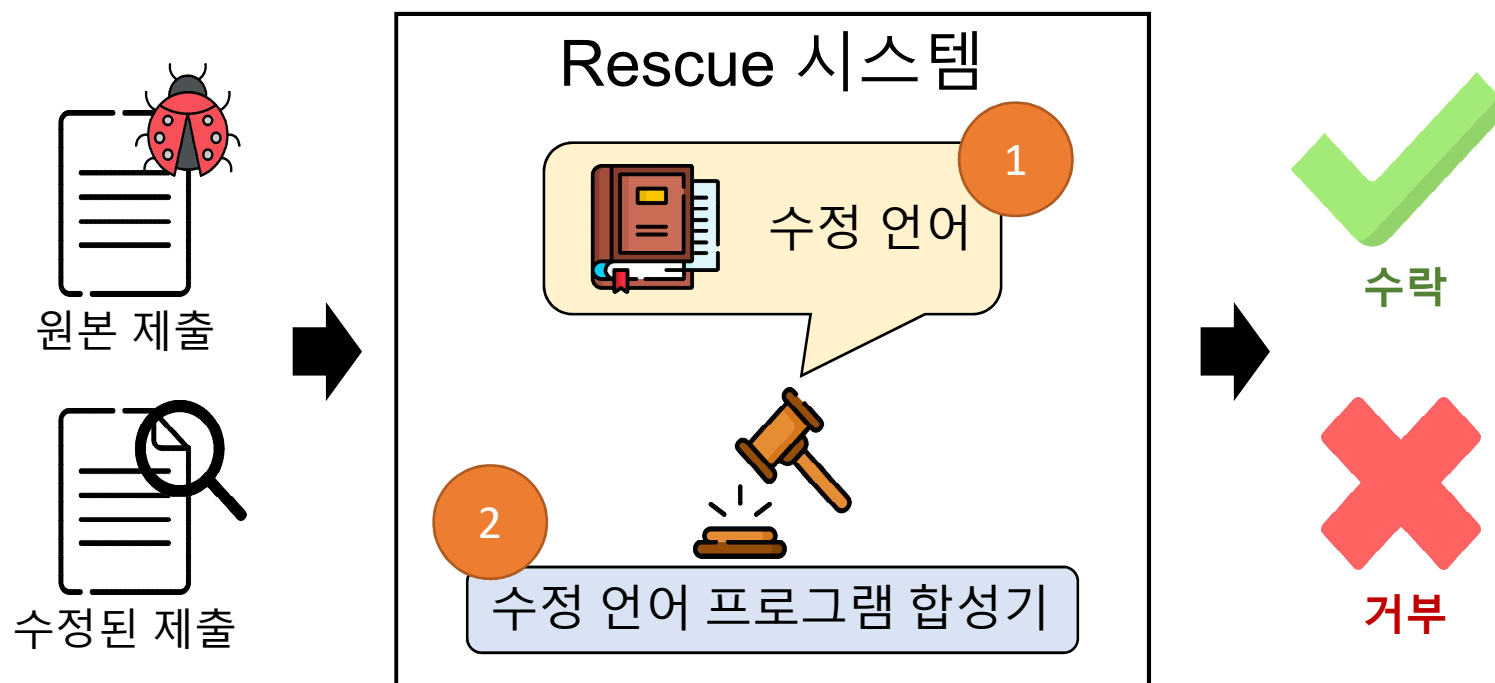
Rescue: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템
 - 수정 언어 프로그램 합성이 성공하면 **승인**
 - 수정 언어 프로그램 합성이 실패하면 **거부**



Rescue: 작은 수정 자동 판별 시스템

- 주어진 수정이 작은 수정인지 판별하는 시스템
 - 수정 언어 프로그램 합성이 성공하면 **승인**
 - 수정 언어 프로그램 합성이 실패하면 **거부**



수정 언어

- 사소한 수정을 정의하는 "수정 언어"

$$Revise : Code \times Cursor \rightarrow Code \times Cursor$$

$$Code = Token^*$$

$$Revise = Cmd^*$$

$$Cmd = \circ \mid \rightarrow \mid \ll \mid \text{Insert}(Str)$$

$$Str = _ \mid " " \mid (\mid) \mid \text{ErrorMessage}$$

수정 언어

- 사소한 수정을 정의하는 “수정 언어”

$Revise : Code \times Cursor \rightarrow Code \times Cursor$

let rec factorial = fun n ->
 if n = 1 then 1
 else n * factorial (n - 1);;

let rec factorial = fun n ->
 if n = 1 then 1
 else n * factorial (n - 1);;

수정 언어

- 사소한 수정을 정의하는 “수정 언어”

$Revise : Code \times Cursor \rightarrow Code \times Cursor$

$Code = Token^*$

```
let rec factorial = fun n ->  
  if n = 1 then 1  
  else n * factorial (n - 1);;
```

```
"let", " ", "rec", " ", "factorial", " ", "=", " ", "fun", " ", "n", " ", "->", "\n",  
" ", " ", "if", " ", "n", " ", "=", " ", "1", " ", " ", "then", " ", " ", "1", " ", "\n", " ", " ",  
"else", " ", "n", " ", " ", "*", " ", " ", "factorial", " ", "(", "n", " ", " ", "-", " ", " ", "1", " ", ")",  
";", ";", ";"
```

수정 언어

- 사소한 수정을 정의하는 "수정 언어"

$Revise : Code \times Cursor \rightarrow Code \times Cursor$

"let", " ", "rec", " ", "factorial", " ", "=", " ", "fun", " ", "n", " ", "->", "\n",
 " ", " ", "if", " ", "n", " ", "=", " ", "1", " ", "then", " ", "1", "\n", " ", " ",
 "else", " ", "n", " ", "*", " ", "factorial", " ", "(", "n", " ", "-", " ", "1", ")",
 ";", ";", ";"

$Revise = Cmd^*$

$Cmd = \circ \mid \rightarrow \mid \ll \mid \text{Insert}(Str)$

$Str = _ \mid " " \mid (\mid) \mid \text{ErrorMessage}$

커서 이동

수정 언어

- 사소한 수정을 정의하는 "수정 언어"

$Revise : Code \times Cursor \rightarrow Code \times Cursor$

"let", " ", "rec", " ", "factorial", " ", "=", " ", "fun", " ", "n", " ", "->", "\n",
 " ", " ", "if", " ", "n", " ", "=", " ", "1", " ", "then", " ", "1", "\n", " ", " ",
 "else", " ", "n", " ", "*", " ", "factorial", " ", "(", "n", " ", "-", " ", "1", ")",
 ";", ";", ";"

$Revise = Cmd^*$

$Cmd = \circ \mid \rightarrow \mid \ll \mid Insert(Str)$

$Str = _ \mid " " \mid (\mid) \mid ErrorMessage$

커서 위치에서 삭제

수정 언어

- 사소한 수정을 정의하는 "수정 언어"

$Revise : Code \times Cursor \rightarrow Code \times Cursor$

"let", " ", "rec", " ", "factorial", " ", "=", " ", "fun", " ", "n", " ", "->", "\n",
 " ", " ", "if", " ", "n", " ", "=", " ", "1", " ", "then", " ", "1", "\n", " ", " ",
 "else", " ", "n", " ", "*", " ", "factorial", " ", "(", "n", " ", "-", " ", "1", ")",
 ";", ";", ";"

$Revise = Cmd^*$

$Cmd = \circ \mid \rightarrow \mid \ll \mid \text{Insert}(Str)$

$Str = _ \mid " " \mid (\mid) \mid \text{ErrorMessage}$

커서 위치에서 Str 삽입

수정 언어

- 사소한 수정을 정의하는 "수정 언어"

$Revise : Code \times Cursor \rightarrow Code \times Cursor$

$Code = Token^*$

$Revise = Cmd^*$

$Cmd = \circ \mid \rightarrow \mid \ll \mid \text{Insert}(Str)$

$Str = _ \mid " " \mid (\mid) \mid \text{ErrorMessage}$

매우 제한된 입력

작은 수정 판별 알고리즘

1. 수정본에서 **삽입할 토큰** 찾기

```
let add3  
= fun a b c ->  
  a + a + c
```

원본 프로그램

```
let add3  
= fun a b c ->  
  a + b + c
```

수정본 프로그램

작은 수정 판별 알고리즘

1. 수정본에서 **삽입할 토큰** 찾기

```
let add3  
= fun a b c ->  
  a + a + c
```

원본 프로그램

```
let add  
= fun a  
  a + b + c
```

23번째 토큰

수정본 프로그램

remove = ..., *add* = ⟨23⟩

작은 수정 판별 알고리즘

2. 삽입할 토큰들의 안전한 삽입을 위한 전처리
 - 예를 들어, 변수명 지우기

```
let add3  
= fun a b c ->  
  a + a + c
```

원본 프로그램

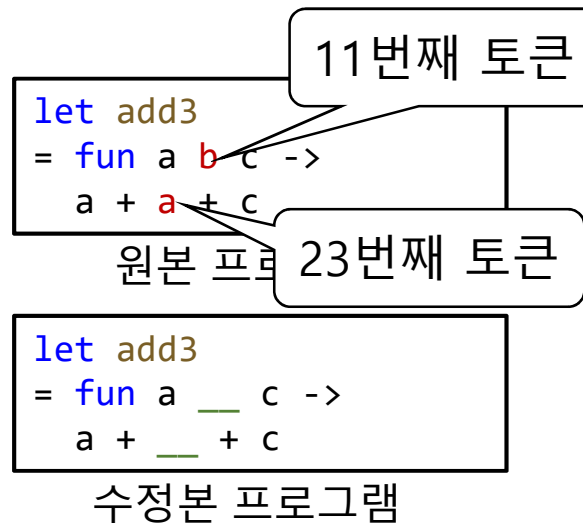
```
let add3  
= fun a _ e  
  a + _ + c
```

b를 _로 대체

수정본 프로그램

작은 수정 판별 알고리즘

3. 원본에서 삭제할 토큰, 수정본에서 삽입할 토큰 찾기



$remove = \langle 11, 23 \rangle, add = \langle 11, 23 \rangle$

작은 수정 판별 알고리즘

4. 수정 언어 프로그램 합성

- 수정 언어 프로그램의 합성에 성공하면 **승인**
- 수정 언어 프로그램의 합성에 실패하면 **거부**

```
let add3  
= fun a b c ->  
  a + a + c
```

원본 프로그램

```
let add3  
= fun a _ c ->  
  a + _ + c
```

수정본 프로그램

remove = $\langle 11, 23 \rangle$, *add* = $\langle 11, 23 \rangle$

Review = ?

수정 언어 프로그램 합성 알고리즘

1. 원본에서 *remove* 토큰을 뒤에서부터 삭제

remove = $\langle 11, 23 \rangle$, *add* = $\langle 11, 23 \rangle$

Reverse = $\langle \quad \rangle$

```
let add3  
= fun a b c ->  
  a + a + c
```

합성된 프로그램의 해석 결과

수정 언어 프로그램 합성 알고리즘

1. 원본에서 *remove* 토큰을 뒤에서부터 삭제

remove = $\langle 11, 23 \rangle$, *add* = $\langle 11, 23 \rangle$

Reverse = $\langle \circ \rightarrow^{23} \ll \rangle$

```
let add3  
= fun a b c ->  
  a + + c
```

합성된 프로그램의 해석 결과

수정 언어 프로그램 합성 알고리즘

1. 원본에서 *remove* 토큰을 뒤에서부터 삭제

$$remove = \langle 11, 23 \rangle, add = \langle 11, 23 \rangle$$

$$Reverse = \langle o \rightarrow^{23} \ll o \rightarrow^{11} \ll \rangle$$

```
let add3  
= fun a c ->  
  a + + c
```

합성된 프로그램의 해석 결과

수정 언어 프로그램 합성 알고리즘

1. 원본에서 *remove* 토큰을 뒤에서부터 삭제
2. 입력이 가능한 *add* 토큰을 앞에서부터 삽입

$$remove = \langle 11, 23 \rangle, add = \langle 11, 23 \rangle$$

$$Result = \langle o \rightarrow^{23} \ll o \rightarrow^{11} \ll \rangle$$

```
let add3  
= fun a c ->  
  a + + c
```

합성된 프로그램의 해석 결과

```
let add3  
= fun a _ c ->  
  a + _ + c
```

수정본 프로그램

수정 언어 프로그램 합성 알고리즘

1. 원본에서 *remove* 토큰을 뒤에서부터 삭제
2. 입력이 가능한 *add* 토큰을 앞에서부터 삽입

remove = $\langle 11, 23 \rangle$, *add* = $\langle 11, 23 \rangle$

Reverse = $\langle \circ \rightarrow^{23} \ll \circ \rightarrow^{11} \ll \circ \rightarrow^{10} \text{insert}(_)^2 \rangle$

```
let add3
= fun a _ c ->
  a + _ + c
```

합성된 프로그램의 해석 결과

```
let add3
= fun a _ c ->
  a + _ + c
```

수정본 프로그램

수정 언어 프로그램 합성 알고리즘

1. 원본에서 *remove* 토큰을 뒤에서부터 삭제
2. 입력이 가능한 *add* 토큰을 앞에서부터 삽입

remove = $\langle 11, 23 \rangle$, *add* = $\langle 11, \boxed{23} \rangle$

Reverse = $\langle \circ \rightarrow^{23} \ll \circ \rightarrow^{11} \ll \circ \rightarrow^{10} \text{insert}(_)^2 \circ \rightarrow^{22} \text{insert}(_)^2 \rangle$

```
let add3
= fun a _ c ->
  a + _ + c
```

합성된 프로그램의 해석 결과

```
let add3
= fun a _ c ->
  a + _ + c
```

수정본 프로그램

수정 언어 프로그램 합성 알고리즘

1. 원본에서 *remove* 토큰을 뒤에서부터 삭제
2. 입력이 가능한 *add* 토큰을 앞에서부터 삽입

$$remove = \langle 11, 23 \rangle, add = \langle 11, 23 \rangle$$

$$Revise = \langle \circ \rightarrow^{23} \ll \circ \rightarrow^{11} \ll \circ \rightarrow^{10} \text{insert}(_)^2 \circ \rightarrow^{22} \text{insert}(_)^2 \rangle$$

```
let add3  
= fun a _ c ->  
  a + _ + c
```

합성된 프로그램의 해석 결과

Rescue 시스템 평가

- FatFinger: 실제 학생의 프로그래밍 실수 모음
 - OCaml: 37개의 사례
 - Python: 45개의 사례

FatFinger 분류

| | 무관한 코드 | 메시지 | 오타 | 사소한 논리 | 심대한 오류 |
|--------|--------|-----|----|--------|--------|
| OCaml | 8 | 13 | 9 | 10 | 3 |
| Python | 0 | 0 | 44 | 1 | 0 |

수락해야 함

거절해야 함

Rescue 시스템 평가

- FatFinger: 실제 학생의 프로그래밍 실수 모음
 - OCaml: 37개의 사례
 - Python: 45개의 사례

FatFinger 분류

| | 무관한 코드 | 메시지 | 오타 | 사소한 논리 | 심대한 오류 |
|--------|--------|-----|----|--------|--------|
| OCaml | 8 | 13 | 9 | 10 | 3 |
| Python | 0 | 0 | 44 | 1 | 0 |

수락해야 함

거절해야 함

- 지금까지 수집된 모든 사례에 대해 정확하게 분류 성공

FatFinger에 대한 정확도

| | OCaml | Python |
|------------|-------|--------|
| Rescue 시스템 | 100% | 100% |

Rescue의 취약점1

1. 수정 언어 설계의 중요성

~~Revise~~ = $Cm\ d^*$

$Cm\ d = M\ ove \mid Backspace \mid Insert(Str)$

$M\ ove = \circ \mid \leftarrow \mid \rightarrow \mid \uparrow \mid \downarrow$

$Str = _ \mid " \mid (\mid) \mid \text{raise ErrorMessage}$

```
--- ocaml/case35/original.ml    2023-01-05 00:36:11.438209375 +0000
+++ ocaml/case35/revised.ml    2023-01-05 01:02:55.280570331 +0000
@@ -168,7 +168,7 @@
     let new_ty = fresh_tyvar () in
     (gen_equations tenv e1 (TyFun (new_ty, ty)))@(gen_equations tenv e2 new_ty)
   | PRINT e -> (gen_equations tenv e ty)
-   | SEQ (e1, e2) -> (gen_equations tenv e2 ty)
+   | SEQ (e1, e2) -> raise TypeError

let solve : typ_eqn -> Subst.t
=fun eqn ->
```

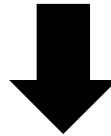
채점용 테스트 케이스에 맞추어 수정

Rescue의 취약점2

2. 악용 가능성

```
let ifthenelse c e1 e2 = if c then e1 else e2
let lt e1 e2 = e1 < e2
let sub e1 e2 = e1 - e2
let mul e1 e2 = e1 * e2
let one = 1;; let two = 2;; ...

let factorial = fun n ->
  TODO TODO TODO TODO TODO TODO TODO TODO TODO ...
```



```
let ifthenelse c e1 e2 = if c then e1 else e2
let lt e1 e2 = e1 < e2
let sub e1 e2 = e1 - e2
let mul e1 e2 = e1 * e2
let one = 1;; let two = 2;; ...

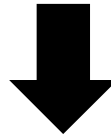
let factorial = fun n ->
  if n < 2 then 1 else n * (factorial (n - 1))
```

Rescue의 취약점2

2. 악용 가능성

```
let ifthenelse c e1 e2 = if c then e1 else e2
let lt e1 e2 = e1 < e2
let sub e1 e2 = e1 - e2
let mul e1 e2 = e1 * e2
let one = 1;; let two = 2;; ...

let factorial = fun n ->
  TODO TODO TODO TODO TODO TODO TODO TODO TODO ...
```



```
let ifthenelse c e1 e2 = if c then e1 else e2
let lt e1 e2 = e1 < e2
let sub e1 e2 = e1 - e2
let mul e1 e2 = e1 * e2
let one = 1;; let two = 2;; ...

let factorial = fun n ->
  ifthenelse (lt n two) one (mul n (factorial (sub n one)))
```

(* 요약 *)

1. 프로그래밍 과제에서 사소한 실수를 하는 것이 흔한 문제이고, 이 때문에 학생과 조교의 스트레스가 많다.
2. 수정이 사소한 수정인지 여부를 자동으로 판별하는 시스템을 제안한다.
 - 사소한 수정이 무엇인지 정의: 수정 언어
 - 자동으로 사소한 수정인지 판별: 수정 언어 프로그램 합성기
3. 실제 사례를 통해 수집된 데이터를 정확하게 분류할 수 있었다.

감사합니다 ☺

FatFinger: 무관한 코드

- 평가하는 구현과 무관한 지점에서 발생하는 오류
 - 예를 들어, 디버깅을 위한 테스트 케이스

```
--- ocaml/case6/original.ml    2022-12-26 02:44:40.901726765 +0000
+++ ocaml/case6/revised.ml     2022-12-26 02:44:37.213778409 +0000
@@ -237,4 +237,4 @@
     if forall check_eq_tyvar subst then Subst.apply t subst
     else raise TypeError;;

-typeof (EQUAL (CONS (CONST 2, NIL), CONS (CONS (CONST 2, NIL), NIL))));;
+typeof (EQUAL (CONS (CONST 2, NIL), CONS (CONS (CONST 2, NIL), NIL))));;
```

테스트 케이스를 실행하다 남겨진 괄호때문에 발생하는 문법 오류

FatFinger: 메시지

- 과제의 명세와 다른 메시지

```
--- ocaml/case21/original.ml 2022-12-28 08:05:33.074647547 +0000
+++ ocaml/case21/revised.ml 2022-12-28 08:05:44.250450447 +0000
@@ -115,7 +115,7 @@
    begin
      match r1, r2 with
      | Int n1, Int n2 ->
-        if n2 = 0 then raise DivisionByZero
+        if n2 = 0 then raise UndefinedSemantics
        else Int (n1 / n2)
      | _ -> raise UndefinedSemantics
    end
```

“UndefinedSemantics”로 예외 처리를 해야 하는 지점에서 임의로 예외 처리

FatFinger: 오타

- 비슷한 다른 변수 / 함수 / 연산자 사용

```
--- ocaml/case31/original.ml 2023-01-02 04:20:19.983240957 +0000
+++ ocaml/case31/revised.ml 2023-01-02 04:21:18.934796635 +0000
@@ -345,7 +345,7 @@
     match v with
     | Num _ | Bool _ | Unit | Record _ ->
       let loc = lookup_loc_env x env in
-       let new_mem = extend_mem (loc, v) mem in
+       let new_mem = extend_mem (loc, v) mem1 in
       (v, new_mem)
     | _ -> raise (UndefinedSemantics)
  end
```

“mem1”대신 같은 타입의 “mem”을 사용해서 발생하는 오류

FatFinger: 사소한 논리

- 새로운 표현식을 도입하지 않고 수정 가능한 오류

```
--- ocaml/case1/original.ml    2022-12-21 08:58:17.038058924 +0000
+++ ocaml/case1/revised.ml    2022-12-21 08:58:23.889968332 +0000
@@ -1,4 +1,4 @@
-let rec exist : int -> 'a list -> bool
+let rec exist
= fun n lst ->
  match lst with
  | [] -> false
```

잘못 명시된 타입때문에 발생하는 오류