Context-Aware and Data-Driven Feedback Generation for Programming Assignments

Dowon Song, Woosuk Lee, and Hakjoo Oh
Korea University and Hanyang University

11 Feb 2022
SW 재난 연구센터 워크샵 @ Jeju

Motivation

- 프로그래밍 교육에 대한 수요가 커지며, 적절한 피드백에 대한 요구가 증가
- 실제 서비스에서도, 피드백의 양과 질이 학생들의 수요를 따라가지 못함

소수구하기 배열에서 자꾸 문제가 생깁니다	1929번 질문	taeyang95
[Python] 최대, 최소 구하기 '틀렸습니다' 질문	10818번 질문	poroli0119
반례가 무엇인지를 모르겠습니다	1158번 질문	hyunsk77
[java]반례를 찾아주십시오..!	1002번 질문	dbtmxlsk2007
연결리스트 시간초과	17298번 질문	y718j
node.js 런타임에러(StackSizeExceeded)	1717번 질문	heech1013
[Python] 반례를 못찾겠습니다 ㅠ 해결	2581번 질문	dnjstjr245
python 10025번 시간초과입니다.	10025번 질문	fblood53
코드에 무슨 이상이 있는 걸까요	2908번 질문	cho990205
[파이썬] 시간초과가 뜨는데 줄이는 방법이 있을까요?	18290번 질문	dhrudgns529
[Python] X보다 작은 수	10871번 질문	poroli0119
(C++) 이분 탐색을 이용한 가장 가까운 수 구하기	질문	o98123
[c++] 계속 틀렸다고 나오는데 어디가 틀렸는지 모르겠어요.. 해결	11022번 질문	hjungwon034
[C++]BFS 코드 반례 질문입니다. 해결	1697번 질문	lcj207
메모리 초과 나는 이유	2887번 질문	chaerin0625
무엇이 틀렸는지 도저히 모르겠습니다,,	10757번 질문	tjswo613
18126 c++ 질문	18126번 질문	fruity
파이썬 메모리 초과 질문입니다	7576번 질문	kms15461
DFS 정상 BFS 시간초과 질문	10026번 질문	codren
데크의 시간복잡도 계산	1158번 질문	nh0903

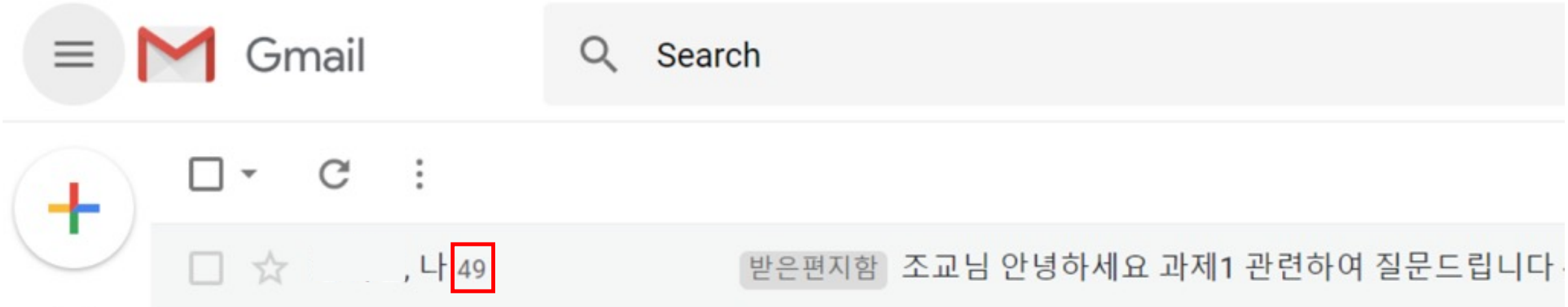
- 온라인 프로그래밍 서비스인 백준의 질의 응답 게시판 상황

- **20개의 질문 중 단 3개**의 질문에 대해서만 피드백이 제공된 상황

자료 출처: 백준 온라인 저지 (https://www.acmicpc.net)

Motivation

- 실제 수업환경에서도 많은 학생들이 과제에 어려움을 느낌
  - 조교가 매 학기 평균 학생들로부터 **100개의 이메일을** 받음
  - 개중 한 질문에 대해서는, **49개의 답변**을 주고받았음



Motivation

- 단순히 모범 답안을 제공하는 것은 학생들에게 도움이 되지 않음

```
let rec do_diff (ae, x) =
  match ae with
  | Const i -> Const 0
  | Var v -> if (v = x) then Const 1 else Const 0
  | Power (v, i) -> if (v = x) Times [Const i; Power (v, i-1)] else Const 0
  | Sum (hd::tl) ->
    if (tl = []) then do_diff (hd, x) else Sum [do_diff (hd, x); do_diff (Sum tl, x)]
  | Times (hd::tl) ->
    if (tl = []) then do_diff (hd, x)
    else Sum [Times ((do_diff (hd, x))::tl); Times [hd; (do_diff (Times tl, x))]]

let rec minimize ae =
  let rec minimize_helper ae =
    match ae with
    | Sum lst ->
      if (lst = []) then Const 0
      else if (List.length lst = 1) then List.hd lst
      else Sum (List.map minimize_helper (List.filter (fun ae -> ae <> Const 0) lst))
    | Times lst ->
      if (lst = []) then Const 0
      else if (List.mem (Const 0)) lst then Const 0
      else if (List.length lst = 1) then List.hd lst
      else Times (List.map minimize_helper (List.filter (fun ae -> ae <> Const 1) lst))
    | _ -> ae in
  let ae' = minimize_helper ae in
  if (ae = ae') then ae else minimize ae'

let diff (ae, str) = minimize (do_diff (ae, str))
```

Student's Submission

```
let rec diff (e, x) =
  match e with
  | Const n -> Const 0
  | Var y -> if (x <> y) then Const 0 else Const 1
  | Power (y, n) -> if (x <> y) then Const 0 else Times [Const n; Power (y, n-1)]
  | Sum (hd::tl) -> Sum (List.map (fun e -> diff (e, x)) (hd::tl))
  | Times [hd] -> diff (hd, x)
  | Times (hd::tl) -> Sum [Times ((diff (hd, x))::tl); Times [hd; diff (Times tl, x)]]
  | _ -> raise (Failure "Invalid")
```

T.A's Solution

Motivation

- 단순히 모범 답안을 제공하는 것은 학생들에게 도움이 되지 않음

```
let rec diff : aexp * string -> aexp
= fun (exp, x) ->
  match exp with
    | Const(i) -> Const(0)
    | Var(s) -> if s=x then Const(1) else Var(s)
    | Power(s, i) ->
      if s=x then if i>2 then Times[Const(i); Power(s, i-1)] else Times[Const(i); Var(s)]
          else Power(s, i)
    | Times(al) ->
      let rec timeiter lst =
        match lst with
          |[] -> []
          |hd::tl ->
           (match hd with
             |Const(i1) -> Const(i1)
             |_ -> diff(hd, x)
           )::(timeiter tl)
      in
      Times(timeiter al)
    | Sum(al) ->
      let rec sumeval lst =
        match lst with
          |[] -> []
          |hd::tl -> if hd=Const(0) then sumeval tl else hd::sumeval tl
      in
      let rec sumiter lst =
        match lst with
          |[] -> []
          |hd::tl -> diff(hd, x)::(sumiter tl)
      in
      Sum(sumeval(sumiter al))
```

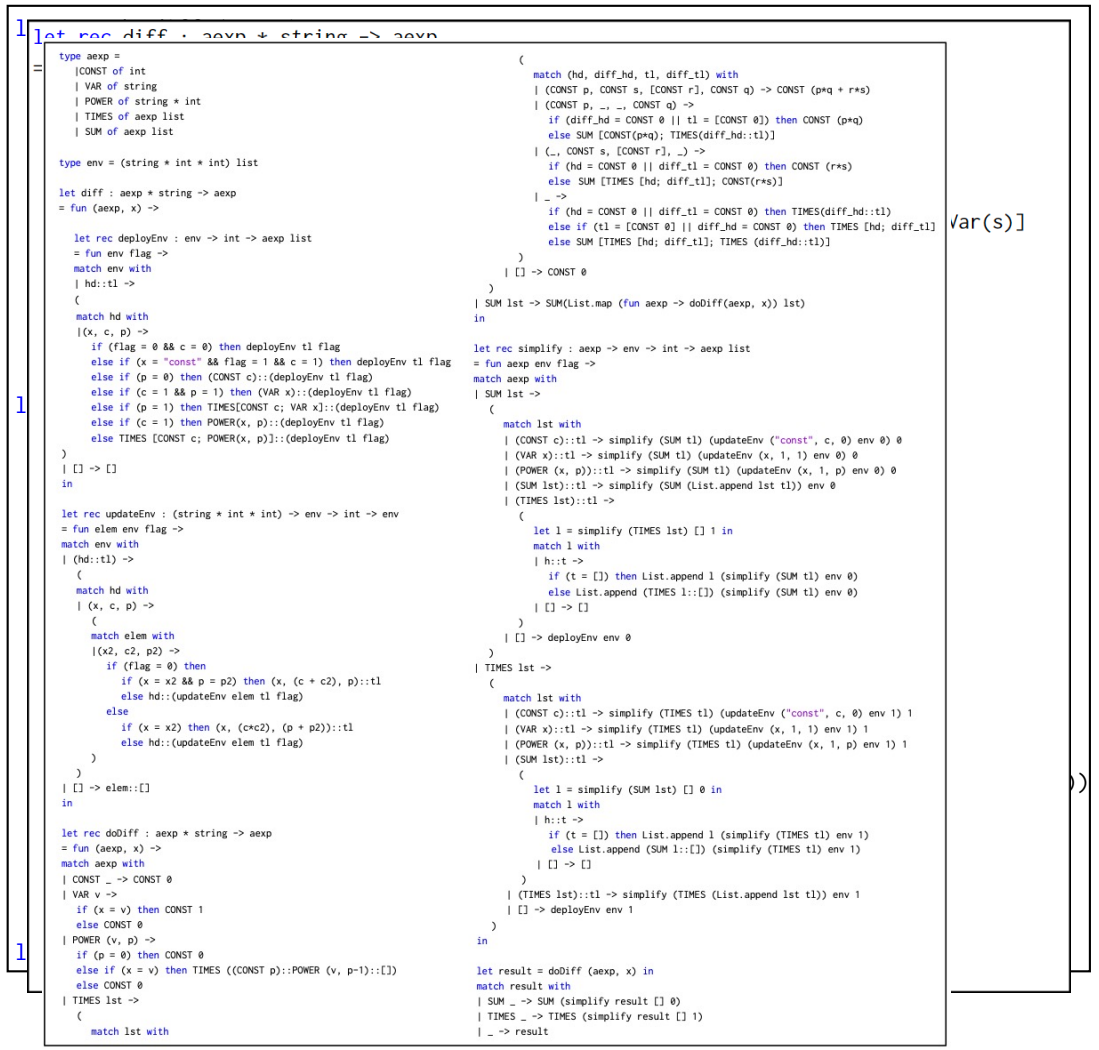
Student's Submission

```
let rec diff (e, x) =
  match e with
  | Const n -> Const 0
  | Var y -> if (x <> y) then Const 0 else Const 1
  | Power (y, n) -> if (x <> y) then Const 0 else Times [Const n; Power (y, n-1)]
  | Sum (hd::tl) -> Sum (List.map (fun e -> diff (e, x)) (hd::tl))
  | Times [hd] -> diff (hd, x)
  | Times (hd::tl) -> Sum [Times ((diff (hd, x))::tl); Times [hd; diff (Times tl, x)]]
  | _ -> raise (Failure "Invalid")
```

T.A's Solution

Motivation

- 단순히 모범 답안을 제공하는 것은 학생들에게 도움이 되지 않음



```
type aexp =
   |CONST of int
   | VAR of string
   | POWER of string * int
   | TIMES of aexp list
   | SUM of aexp list

type env = (string * int * int) list

let diff : aexp * string -> aexp
= fun (aexp, x) ->

   let rec deployEnv : env -> int -> aexp list
   = fun env flag ->
   match env with
   | hd::tl ->
   (
   match hd with
   |(x, c, p) ->
      if (flag = 0 && c = 0) then deployEnv tl flag
      else if (x = "const" && flag = 1 && c = 1) then deployEnv tl flag
      else if (p = 0) then (CONST c)::(deployEnv tl flag)
      else if (c = 1 && p = 1) then (VAR x)::(deployEnv tl flag)
      else if (p = 1) then TIMES[CONST c; VAR x]::(deployEnv tl flag)
      else if (c = 1) then POWER(x, p)::(deployEnv tl flag)
      else TIMES [CONST c; POWER(x, p)]::(deployEnv tl flag)
   )
   | [] -> []
   in

   let rec updateEnv : (string * int * int) -> env -> int -> env
   = fun elem env flag ->
   match env with
   | (hd::tl) ->
      (
      match hd with
      | (x, c, p) ->
         (
         match elem with
         |(x2, c2, p2) ->
            if (flag = 0) then
               if (x = x2 && p = p2) then (x, (c + c2), p)::tl
               else hd::(updateEnv elem tl flag)
            else
               if (x = x2) then (x, (c*c2), (p + p2))::tl
               else hd::(updateEnv elem tl flag)
         )
      )
   | [] -> elem::[]
   in

   let rec doDiff : aexp * string -> aexp
   = fun (aexp, x) ->
   match aexp with
   | CONST _ -> CONST 0
   | VAR v ->
      if (x = v) then CONST 1
      else CONST 0
   | POWER (v, p) ->
      if (p = 0) then CONST 0
      else if (x = v) then TIMES ((CONST p)::POWER (v, p-1)::[])
      else CONST 0
   | TIMES lst ->
      (
         match lst with
         (
            match (hd, diff_hd, tl, diff_tl) with
            | (CONST p, CONST s, [CONST r], CONST q) -> CONST (p*q + r*s)
            | (CONST p, _, _, CONST q) ->
               if (diff_hd = CONST 0 || tl = [CONST 0]) then CONST (p*q)
               else SUM [CONST(p*q); TIMES(diff_hd::tl)]
            | (_, CONST s, [CONST r], _) ->
               if (hd = CONST 0 || diff_tl = CONST 0) then CONST (r*s)
               else SUM [TIMES [hd; diff_tl]; CONST(r*s)]
            | _ ->
               if (hd = CONST 0 || diff_tl = CONST 0) then TIMES(diff_hd::tl)
               else if (tl = [CONST 0] || diff_hd = CONST 0) then TIMES [hd; diff_tl]
               else SUM [TIMES [hd; diff_tl]; TIMES (diff_hd::tl)]
         )
      | [] -> CONST 0
   )
   | SUM lst -> SUM(List.map (fun aexp -> doDiff(aexp, x)) lst)
   in

   let rec simplify : aexp -> env -> int -> aexp list
   = fun aexp env flag ->
   match aexp with
   | SUM lst ->
      (
         match lst with
         | (CONST c)::tl -> simplify (SUM tl) (updateEnv ("const", c, 0) env 0) 0
         | (VAR x)::tl -> simplify (SUM tl) (updateEnv (x, 1, 1) env 0) 0
         | (POWER (x, p))::tl -> simplify (SUM tl) (updateEnv (x, 1, p) env 0) 0
         | (SUM lst)::tl -> simplify (SUM (List.append lst tl)) env 0
         | (TIMES lst)::tl ->
            (
               let l = simplify (TIMES lst) [] 1 in
               match l with
               | h::t ->
                  if (t = []) then List.append l (simplify (SUM tl) env 0)
                  else List.append (TIMES l::[]) (simplify (SUM tl) env 0)
               | [] -> []
            )
         | [] -> deployEnv env 0
      )
   | TIMES lst ->
      (
         match lst with
         | (CONST c)::tl -> simplify (TIMES tl) (updateEnv ("const", c, 0) env 1) 1
         | (VAR x)::tl -> simplify (TIMES tl) (updateEnv (x, 1, 1) env 1) 1
         | (POWER (x, p))::tl -> simplify (TIMES tl) (updateEnv (x, 1, p) env 1) 1
         | (SUM lst)::tl ->
            (
               let l = simplify (SUM lst) [] 0 in
               match l with
               | h::t ->
                  if (t = []) then List.append l (simplify (TIMES tl) env 1)
                  else List.append (SUM l::[]) (simplify (TIMES tl) env 1)
               | [] -> []
            )
         | (TIMES lst)::tl -> simplify (TIMES (List.append lst tl)) env 1
         | [] -> deployEnv env 1
      )
   in

   let result = doDiff (aexp, x) in
   match result with
   | SUM _ -> SUM (simplify result [] 0)
   | TIMES _ -> TIMES (simplify result [] 1)
   | _ -> result
```

```
let rec diff (e, x) =
  match e with
  | Const n -> Const 0
  | Var y -> if (x <> y) then Const 0 else Const 1
  | Power (y, n) -> if (x <> y) then Const 0 else Times [Const n; Power (y, n-1)]
  | Sum (hd::tl) -> Sum (List.map (fun e -> diff (e, x)) (hd::tl))
  | Times [hd] -> diff (hd, x)
  | Times (hd::tl) -> Sum [Times ((diff (hd, x))::tl); Times [hd; diff (Times tl, x)]]
  | _ -> raise (Failure "Invalid")
```

T.A's Solution

Motivation

- 단순히 모범 답안을 제공하는 것은 학생들에게 도움이 되지 않음

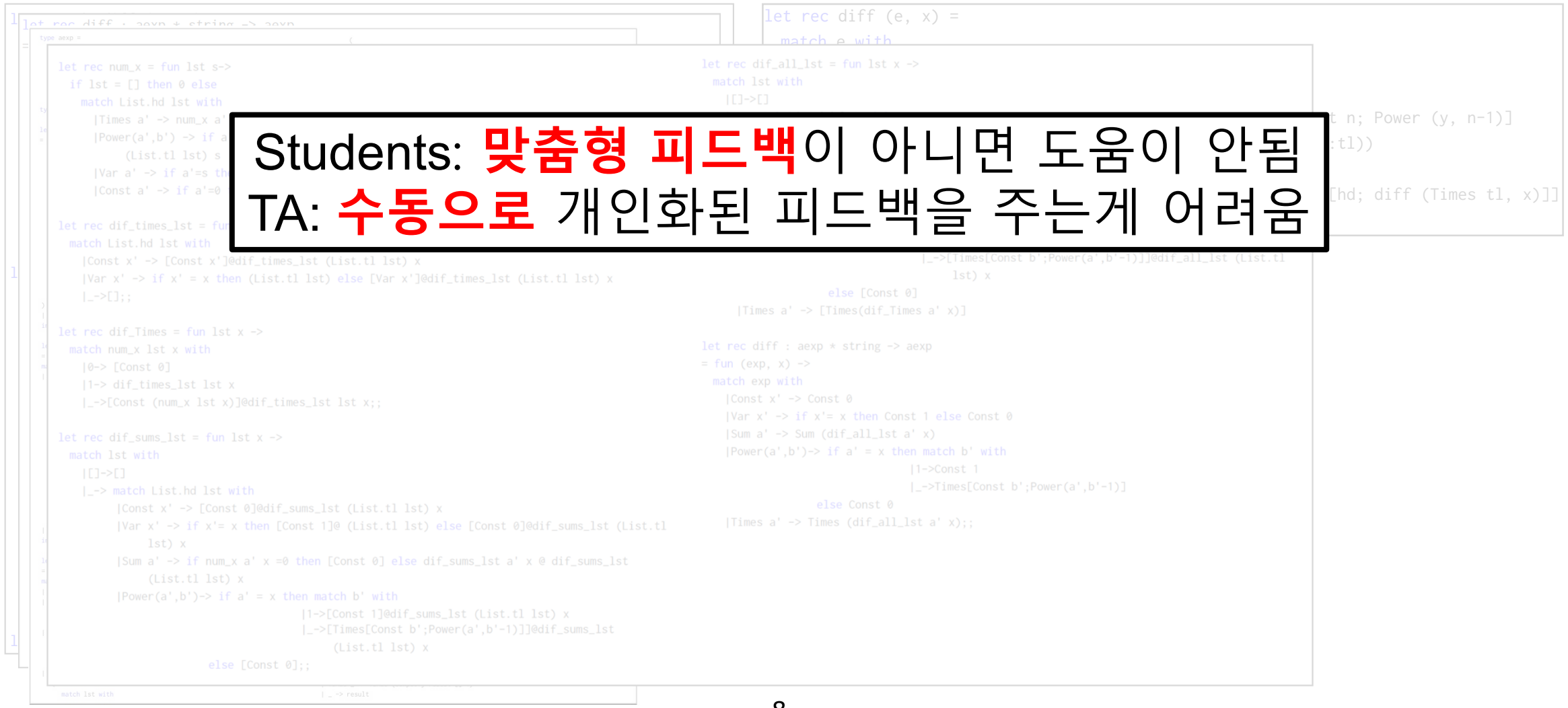
```
let rec num_x = fun lst s->
  if lst = [] then 0 else
    match List.hd lst with
      |Times a' -> num_x a' s + num_x (List.tl lst) s
      |Power(a',b') -> if a' = s && b'<> 0 then b' + num_x (List.tl lst) s else 0 +num_x
           (List.tl lst) s
      |Var a' -> if a'=s then 1 + num_x (List.tl lst) s else 0 +num_x (List.tl lst) s
      |Const a' -> if a'=0 then 0 else 0 +num_x (List.tl lst) s;;

let rec dif_times_lst = fun lst x ->
  match List.hd lst with
    |Const x' -> [Const x']@dif_times_lst (List.tl lst) x
    |Var x' -> if x' = x then (List.tl lst) else [Var x']@dif_times_lst (List.tl lst) x
    |_->[];;

let rec dif_Times = fun lst x ->
  match num_x lst x with
    |0-> [Const 0]
    |1-> dif_times_lst lst x
    |_->[Const (num_x lst x)]@dif_times_lst lst x;;

let rec dif_sums_lst = fun lst x ->
  match lst with
    |[]->[]
    |_-> match List.hd lst with
         |Const x' -> [Const 0]@dif_sums_lst (List.tl lst) x
         |Var x' -> if x'= x then [Const 1]@ (List.tl lst) else [Const 0]@dif_sums_lst (List.tl
              lst) x
         |Sum a' -> if num_x a' x =0 then [Const 0] else dif_sums_lst a' x @ dif_sums_lst
              (List.tl lst) x
         |Power(a',b')-> if a' = x then match b' with
                                    |1->[Const 1]@dif_sums_lst (List.tl lst) x
                                    |_->[Times[Const b';Power(a',b'-1)]]@dif_sums_lst
                                         (List.tl lst) x
                        else [Const 0];;
```

```
let rec dif_all_lst = fun lst x ->
  match lst with
    |[]->[]
    |_->match List.hd lst with
      |Const x' -> [Const 0]@dif_all_lst (List.tl lst) x
      |Var x' -> if x'= x then [Const 1]@ (List.tl lst) else [Const 0]@dif_all_lst (List.tl
           lst) x
      |Sum a' -> if num_x a' x =0 then [Const 0] else dif_sums_lst a' x @ dif_all_lst (List.tl
           lst) x
      |Power(a',b')-> if a' = x then match b' with
                                    |1->[Const 1]@dif_all_lst (List.tl lst) x
                                    |_->[Times[Const b';Power(a',b'-1)]]@dif_all_lst (List.tl
                                         lst) x
                     else [Const 0]
      |Times a' -> [Times(dif_Times a' x)]

let rec diff : aexp * string -> aexp
= fun (exp, x) ->
  match exp with
    |Const x' -> Const 0
    |Var x' -> if x'= x then Const 1 else Const 0
    |Sum a' -> Sum (dif_all_lst a' x)
    |Power(a',b')-> if a' = x then match b' with
                                   |1->Const 1
                                   |_->Times[Const b';Power(a',b'-1)]
                   else Const 0
    |Times a' -> Times (dif_all_lst a' x);;
```

```
let rec diff (e, x) =
  match e with
                                                    nst n; Power (y, n-1)]
                                                    d::tl))

                                                    s [hd; diff (Times tl, x)]]
```

Motivation

- 단순히 모범 답안을 제공하는 것은 학생들에게 도움이 되지 않음



Motivation

- 단순히 모범 답안을 제공하는 것은 학생들에게 도움이 되지 않음

Students: **맞춤형 피드백**이 아니면 도움이 안됨
TA: **수동으로** 개인화된 피드백을 주는게 어려움

Providing **Personalized** Feedback **Automatically**

Data-Driven Feedback Generation

- 최근의 연구 동향은 **많은 정답 코드**를 활용하여 피드백을 생성. e.g., CLARA (PLDI 18), SARFGEN (PLDI 18)

Data-Driven Feedback Generation

- 최근의 연구 동향은 **많은 정답 코드**를 활용하여 피드백을 생성. e.g., CLARA (PLDI 18), SARFGEN (PLDI 18)

- 이러한 연구들은 주어진 오답과 **아주 비슷한** 정답 프로그램의 존재를 가정함. e.g., exact CFG matching

Data-Driven Feedback Generation

- 최근의 연구 동향은 **많은 정답 코드**를 활용하여 피드백을 생성. e.g., CLARA (PLDI 18), SARFGEN (PLDI 18)

- 이러한 연구들은 주어진 오답과 **아주 비슷한** 정답 프로그램의 존재를 가정함. e.g., exact CFG matching

- 이런 강한 가정은 **초급 난이도의 문제를 벗어나면** 거의 만족되지 않음.

Difficulty	# Submissions	# Error	# Matching	Matching Rate
Introductory	1,185	105	87	83%
Intermediate	1,107	116	42	**36%**
Advanced	1,919	443	80	**18%**

Motivating Example

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      (match h with
5      | Int n -> (Int n)::timediff(t,x)
6      | Var v -> if v=x then (Int 1)::timediff(t,x)
7                 else (Int 0)::timediff(t,x)
8      | Pow (v,c) ->
9        if v=x then (Mul [Int n;Pow(v,c-1)])::timediff(t,x)
10       else (Int 0)::timediff(t,x))

11 and sumdiff (slst,x) =
12   match slst with
13   | [] -> [] | h::t ->
14     (match h with
15     | Int n -> (Int 0)::sumdiff(t,x)
16     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
17                else (Int 0)::sumdiff(t,x)
18     | Pow (v,c) ->
19       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
20       else (Int 0)::sumdiff(t,x)
21     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
22     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))

23 and diff (e, x) =
24   match e with
25   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
26   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
27   | Sum lst -> Sum (sumdiff(lst,x))
28   | Mul lst -> Mul (timediff(lst,x))
```

- 3가지의 함수를 사용해 작성된 오답 코드
  - 세개의 함수 모두 오류가 존재
  - 최소로 수정하기 위해선 각 함수를 수정해야함

Motivating Example

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      (match h with
5      | Int n -> (Int n)::timediff(t,x)
6      | Var v -> if v=x then (Int 1)::timediff(t,x)
7                 else (Int 0)::timediff(t,x)
8      | Pow (v,c) ->
9        if v=x then (Mul [Int n;Pow(v,c-1)])::timediff(t,x)
10       else (Int 0)::timediff(t,x))

11 and sumdiff (slst,x) =
12   match slst with
13   | [] -> [] | h::t ->
14     (match h with
15     | Int n -> (Int 0)::sumdiff(t,x)
16     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
17                else (Int 0)::sumdiff(t,x)
18     | Pow (v,c) ->
19       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
20       else (Int 0)::sumdiff(t,x)
21     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
22     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))

23 and diff (e, x) =
24   match e with
25   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
26   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
27   | Sum lst -> Sum (sumdiff(lst,x))
28   | Mul lst -> Mul (timediff(lst,x))
```

- 3가지의 함수를 사용해 작성된 오답 코드
  - 세개의 함수 모두 오류가 존재
  - 최소로 수정하기 위해선 각 함수를 수정해야함

- 218개의 정답 코드 중 **같은 CFG구조**가 하나도 없음

Motivating Example

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      (match h with
5      | Int n -> (Int n)::timediff(t,x)
6      | Var v -> if v=x then (Int 1)::timediff(t,x)
7                 else (Int 0)::timediff(t,x)
8      | Pow (v,c) ->
9        if v=x then (Mul [Int n;Pow(v,c-1)])::timediff(t,x)
10       else (Int 0)::timediff(t,x))

11 and sumdiff (slst,x) =
12   match slst with
13   | [] -> [] | h::t ->
14     (match h with
15     | Int n -> (Int 0)::sumdiff(t,x)
16     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
17                else (Int 0)::sumdiff(t,x)
18     | Pow (v,c) ->
19       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
20       else (Int 0)::sumdiff(t,x)
21     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
22     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))

23 and diff (e, x) =
24   match e with
25   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
26   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
27   | Sum lst -> Sum (sumdiff(lst,x))
28   | Mul lst -> Mul (timediff(lst,x))
```

- 3가지의 함수를 사용해 작성된 오답 코드
  - 세개의 함수 모두 오류가 존재
  - 최소로 수정하기 위해선 각 함수를 수정해야함
- 218개의 정답 코드 중 **같은 CFG구조**가 하나도 없음
- 하지만 해당 프로그램과 **부분적으로 비슷한** 정답은 존재함

Motivating Example

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      (match h with
5      | Int n -> (Int n)::timediff(t,x)
6      | Var v -> if v=x then (Int 1)::timediff(t,x)
7                 else (Int 0)::timediff(t,x)
8      | Pow (v,c) ->
9        if v=x then (Mul [Int n;Pow(v,c-1)])::timediff(t,x)
10       else (Int 0)::timediff(t,x))

11 and sumdiff (slst,x) =
12   match slst with
13   | [] -> [] | h::t ->
14     (match h with
15     | Int n -> (Int 0)::sumdiff(t,x)
16     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
17                else (Int 0)::sumdiff(t,x)
18     | Pow (v,c) ->
19       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
20       else (Int 0)::sumdiff(t,x)
21     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
22     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))

23 and diff (e, x) =
24   match e with
25   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
26   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
27   | Sum lst -> Sum (sumdiff(lst,x))
28   | Mul lst -> Mul (timediff(lst,x))
```

```
1  let rec check (l,v) =
2    match l with
3    | [] -> false | h::t ->
4      (match h with
5      | Var x -> if x=v then true else check(t,v)
6      | Pow (x,i) -> if x=v then true else check(t,v)
7      | Sum lst | Mul lst -> check(lst,v) || check(t,v)
8      | _ -> check(t,v))

9  let rec times (l,v) =
10   match l with
11   | [] -> [] | h::t ->
12     if t=[] then [diff(h,v)]
13     else [Mul ([diff(h,v)]@t)]@[Mul ([h]@[Sum (times(t,v))])]

14 and sum (l,v) =
15   match l with
16   | [] -> [] | h::t -> [diff(h,v)]@(sum(t,v))

17 and diff (aexp,v) =
18   match aexp with
19   | Int n -> Int 0 | Var x -> if x=v then Int 1 else Int 0
20   | Pow (s,i) ->
21     if s=str then (
22       if i=1 then Int 1
23       else if i=0 then Int 0
24       else Mul ([Int i; Pow (s,i-1)]))
25     else Int 0
26   | Mul l -> if check(l,v) then Sum (times(l,v)) else Int 0
27   | Sum l -> Sum (sum (l,v))
```

Motivating Example

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      (match h with
5      | Int n -> (Int n)::timediff(t,x)
6      | Var v -> if v=x then (Int 1)::timediff(t,x)
7                 else (Int 0)::timediff(t,x)
8      | Pow (v,c) ->
9        if v=x then (Mul [Int n;Pow(v,c-1)])::timediff(t,x)
10       else (Int 0)::timediff(t,x))

11 and sumdiff (slst,x) =
12   match slst with
13   | [] -> [] | h::t ->
14     (match h with
15     | Int n -> (Int 0)
16     | Var v -> if v=x t
17                else (In
18     | Pow (v,c) ->
19       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
20       else (Int 0)::sumdiff(t,x)
21     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
22     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))

23 and diff (e, x) =
24   match e with
25   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
26   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
27   | Sum lst -> Sum (sumdiff(lst,x))
28   | Mul lst -> Mul (timediff(lst,x))
```

```
1  let rec check (l,v) =
2    match l with
3    | [] -> false | h::t ->
4      (match h with
5      | Var x -> if x=v then true else check(t,v)
6      | Pow (x,i) -> if x=v then true else check(t,v)
7      | Sum lst | Mul lst -> check(lst,v) || check(t,v)
8      | _ -> check(t,v))

9  let rec times (l,v) =
10   match l with
11   | [] -> [] | h::t ->
12     if t=[] then [diff(h,v)]
13     else [Mul ([diff(h,v)]@t)]@[Mul ([h]@[Sum (times(t,v))])]

14 and sum (l,v) =
                                    sum(t,v))
17 and diff (aexp,v) =
18   match aexp with
19   | Int n -> Int 0 | Var x -> if x=v then Int 1 else Int 0
20   | Pow (s,i) ->
21     if s=str then (
22       if i=1 then Int 1
23       else if i=0 then Int 0
24       else Mul ([Int i; Pow (s,i-1)]))
25     else Int 0
26   | Mul l -> if check(l,v) then Sum (times(l,v)) else Int 0
27   | Sum l -> Sum (sum (l,v))
```

오답 코드에는 없는 다른 함수들을 사용함

Motivating Example

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      (match h with
5      | Int n -> (Int n)::timediff(t,x)
6      | Var v -> if v=x then (Int 1)::timediff(t,x)
7                 else (Int 0)::timediff(t,x)
8      | Pow (v,c) ->
9        if v=x then (Mul [Int n;Pow(v,c-1)])::timediff(t,x)
10       else (Int 0)::timediff(t,x))
11 and sumdiff (slst,x) =
12   match slst with
13   | [] -> [] | h::t ->
14     (match h with
15     | Int n -> (Int 0)
16     | Var v -> if v=x
17                else (I
18     | Pow (v,c) ->
19       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
20       else (Int 0)::sumdiff(t,x)
21     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
22     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))
23 and diff (e, x) =
24   match e with
25   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
26   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
27   | Sum lst -> Sum (sumdiff(lst,x))
28   | Mul lst -> Mul (timediff(lst,x))
```

```
1  let rec check (l,v) =
2    match l with
3    | [] -> false | h::t ->
4      (match h with
5      | Var x -> if x=v then true else check(t,v)
6      | Pow (x,i) -> if x=v then true else check(t,v)
7      | Sum lst | Mul lst -> check(lst,v) || check(t,v)
8      | _ -> check(t,v))
9  let rec times (l,v) =
10   match l with
11   | [] -> [] | h::t ->
12     if t=[] then [diff(h,v)]
13     else [Mul ([diff(h,v)]@t)]@[Mul ([h]@[Sum (times(t,v))])]
14 and sum (l,v) =
                                                    um(t,v))
17 and diff (aexp,v) =
18   match aexp with
19   | Int n -> Int 0 | Var x -> if x=v then Int 1 else Int 0
20   | Pow (s,i) ->
21     if s=str then (
22       if i=1 then Int 1
23       else if i=0 then Int 0
24       else Mul ([Int i; Pow (s,i-1)]))
25     else Int 0
26   | Mul l -> if check(l,v) then Sum (times(l,v)) else Int 0
27   | Sum l -> Sum (sum (l,v))
```

오류 함수와 비슷한 기능을 갖는 함수 존재

Our Approach: CAFE

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      if t=[] then [diff(h,v)] else
5        [Mul ([diff(h,x)]@t)]@[Mul ([h]@[Sum (timediff(t,x))])]
6  and sumdiff (slst,x) =
7    match slst with
8    | [] -> [] | h::t ->
9      (match h with
10     | Int n -> (Int 0)::sumdiff(t,x)
11     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
12                else (Int 0)::sumdiff(t,x)
13     | Pow (v,c) ->
14       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
15       else (Int 0)::sumdiff(t,x)
16     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
17     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))
18 and diff (e, x) =
19   match e with
20   | Int n -> Int 0 |
21   | Pow (v,c) -> if
22   | Sum lst -> Sum (
23   | Mul lst -> Mul (timediff(lst,x))
```

```
1  let rec check (l,v) =
2    match l with
3    | [] -> false | h::t ->
4      (match h with
5      | Var x -> if x=v then true else check(t,v)
6      | Pow (x,i) -> if x=v then true else check(t,v)
7      | Sum lst | Mul lst -> check(lst,v) || check(t,v)
8      | _ -> check(t,v))
9  let rec times (l,v) =
10   match l with
11   | [] -> [] | h::t ->
12     if t=[] then [diff(h,v)]
13     else [Mul ([diff(h,v)]@t)]@[Mul ([h]@[Sum (times(t,v))])]
14 and sum (l,v) =
15   match l with
16   | [] -> [] | h::t -> [diff(h,v)]@(sum(t,v))
17 and diff (aexp,v) =
18   match aexp with
                                    en Int 1 else Int 0
22       if i=1 then Int 1
23       else if i=0 then Int 0
24       else Mul ([Int i; Pow (s,i-1)]))
25     else Int 0
26   | Mul l -> if check(l,v) then Sum (times(l,v)) else Int 0
27   | Sum l -> Sum (sum (l,v))
```

CAFE: **해당 함수만을 이용해** 오류 함수 수정

Our Approach: CAFE

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      if t=[] then [diff(h,v)] else
5        [Mul ([diff(h,x)]@t)]@[Mul ([h]@[Sum (timediff(t,x))])]
6  and sumdiff (slst,x) =
7    match slst with
8    | [] -> [] | h::t ->
9      (match h with
10     | Int n -> (Int 0)::sumdiff(t,x)
11     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
12                else (Int 0)::sumdiff(t,x)
13     | Pow (v,c) ->
14       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
15       else (Int 0)::sumdiff(t,x)
16     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
17     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))
18 and diff (e, x) =
19   match e with
20   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
21   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
22   | Sum lst -> Sum (sumdiff(lst,x))
23   | Mul lst -> Mul (timediff(lst,x))
```

```
1  let rec length lst = List.length lst
2  let rec diff_sum (l, k) = match l with
3    | [] -> [] | h::t -> (diff(h,k))::(diff_sum(t,k))
4  let rec f (l, k, p, q) = match l with
5    | [] -> [] | h::t ->
6      if p=q then diff(h,k)::f(t,k,p+1,q)
7      else hd::(f(t,k,p+1,q))
8  let rec diff_time (l, k, x, y) =
9    if x>y then [] else Mul (f(l,k,1,x))::diff_time(l,k,x+1,y)
10 let rec diff (e, x) = match e with
11   | Sum lst -> Sum (diff_sum(lst,x))
12   | Mul lst -> Sum (diff_time(lst,x,1,(length lst))) | …
```

Our Approach: CAFE

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      if t=[] then [diff(h,v)] else
5        [Mul ([diff(h,x)]@t)]@[Mul ([h]@[Sum (timediff(t,x))])]
6  and sumdiff (slst,x) =
7    match slst with
8    | [] -> [] | h::t ->
9      (match h with
10     | Int n -> (Int 0)::sumdiff(t,x)
11     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
12                else (Int 0)::sumdiff(t,x)
13     | Pow (v,c) ->
14       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
15       else (Int 0)::sumdiff(t,x)
16     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
17     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))
18 and diff (e, x) =
19   match e with
20   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
21   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
22   | Sum lst -> Sum (sumdiff(lst,x))
23   | Mul lst -> Mul (timediff(lst,x))
```

```
1  let rec length lst = List.length lst
2  let rec diff_sum (l, k) = match l with
3    | [] -> [] | h::t -> (diff(h,k))::(diff_sum(t,k))
4  let rec f (l, k, p, q) = match l with
5    | [] -> [] | h::t ->
6      if p=q then diff(h,k)::f(t,k,p+1,q)
7      else hd::(f(t,k,p+1,q))
8  let rec diff_time (l, k, x, y) =
9    if x>y then [] else Mul (f(l,k,1,x))::diff_time(l,k,x+1,y)
10 let rec diff (e, x) = match e with
11   | Sum lst -> Sum (diff_sum(lst,x))
12   | Mul lst -> Sum (diff_time(lst,x,1,(length lst))) | …
```

Our Approach: CAFE

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      if t=[] then [diff(h,v)] else
5        [Mul ([diff(h,x)]@t)]@[Mul ([h]@[Sum (timediff(t,x))])]
6  and sumdiff (slst,x) =
7    match slst with
8    | [] -> [] | h::t ->
9      (match h with
10     | Int n -> (Int 0)::sumdiff(t,x)
11     | Var v -> if v=x then (Int 1)::sumdiff(t,x)
12                else (Int 0)::sumdiff(t,x)
13     | Pow (v,c) ->
14       if v=x then (Mul [Int n;Pow(v,c-1)])::sumdiff(t,x)
15       else (Int 0)::sumdiff(t,x)
16     | Sum lst -> Sum (sumdiff (lst,x))::sumdiff(t,x)
17     | Time lst -> Mul (timediff (lst,x))::sumdiff(t,x))
18 and diff (e, x) =
19   match e with
20   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
21   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
22   | Sum lst -> Sum (sumdiff(lst,x))
23   | Mul lst -> Mul (timediff(lst,x))
```

```
1  let rec length lst = List.length lst
2  let rec diff_sum (l, k) = match l with
3    | [] -> [] | h::t -> (diff(h,k))::(diff_sum(t,k))
4  let rec f (l, k, p, q) = match l with
5    | [] -> [] | h::t ->
6      if p=q then diff(h,k)::f(t,k,p+1,q)
7      else hd::(f(t,k,p+1,q))
8  let rec diff_time (l, k, x, y) =
9    if x>y then [] else Mul (f(l,k,1,x))::diff_time(l,k,x+1,y)
10 let rec diff (e, x) = match e with
11   | Sum lst -> Sum (diff_sum(lst,x))
12   | Mul lst -> Sum (diff_time(lst,x,1,(length lst))) | …
```

```
1  let rec differ (a, x) =
2    match a with
3    | Int a -> Int 0 | Var v -> if x=v then Int 1 else Int 0
4    | Pow (v, a) ->
5      if x=v then Mul ([Int a]@[Pow (v,a-1)]) else Int 0
6    | Sum l -> Sum (sum(l,x)) | Mul l -> Sum (times(l,x,[]))
7  and times (a, x, acc) =
8    match a with
9    | [] -> [] | hd::tl ->
10     if List.length a = List.length acc then acc
11     else (times((tl@[hd]),x,acc@[Mul ([diff(hd,x)]@tl)]))
12 and sum (a, x) = match a with
13   | [] -> [] | hd::tl -> ([(diff(hd,x))]@(sum(tl,x)))
14 let rec diff (a, x) = differ(a,x)
```

Our Approach: CAFE

```
1  let rec timediff (tst,x) =
2    match tst with
3    | [] -> [] | h::t ->
4      if t=[] then [diff(h,v)] else
5        [Mul ([diff(h,x)]@t)]@[Mul ([h]@[Sum (timediff(t,x))])]
6  and sumdiff (slst,x) =
7    match slst with
8    | [] -> [] | h::t ->
       (diff(hd,key))::(sumdiff(t,x))
9  and diff (e, x) =
10   match e with
11   | Int n -> Int 0 | Var v -> if v=x then Int 1 else Int 0
12   | Pow (v,c) -> if v=x then Mul [Int n;Pow(v,c-1)] else Int 0
13   | Sum lst -> Sum (sumdiff(lst,x))
14   | Mul lst -> Sum (timediff(lst,x))
```

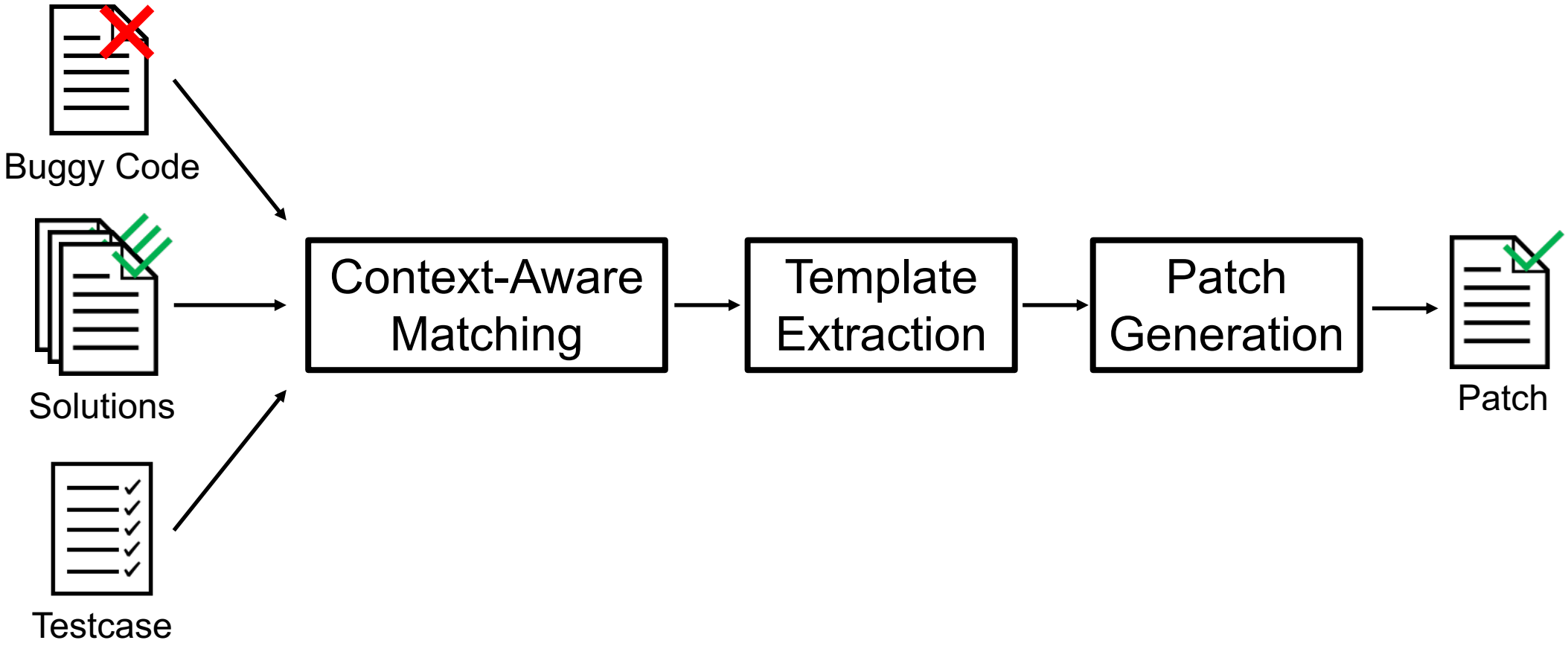
```
1  let rec length lst = List.length lst
2  let rec diff_sum (l, k) = match l with
3    | [] -> [] | h::t -> (diff(h,k))::(diff_sum(t,k))
4  let rec f (l, k, p, q) = match l with
5    | [] -> [] | h::t ->
6      if p=q then diff(h,k)::f(t,k,p+1,q)
7      else hd::(f(t,k,p+1,q))
8  let rec diff_time (l, k, x, y) =
9    if x>y then [] else Mul (f(l,k,1,x))::diff_time(l,k,x+1,y)
10 let rec diff (e, x) = match e with
11   | Sum lst -> Sum (diff_sum(lst,x))
12   | Mul lst -> Sum (diff_time(lst,x,1,(length lst))) | …
```

```
1  let rec differ (a, x) =
2    match a with
3    | Int a -> Int 0 | Var v -> if x=v then Int 1 else Int 0
4    | Pow (v, a) ->
5      if x=v then Mul ([Int a]@[Pow (v,a-1)]) else Int 0
6    | Sum l -> Sum (sum(l,x)) | Mul l -> Sum (times(l,x,[]))
7  and times (a, x, acc) =
8    match a with
9    | [] -> [] | hd::tl ->
10     if List.length a = List.length acc then acc
                                          x)]@tl)]))
                                          tl,x)))
14 let rec diff (a, x) = differ(a,x)
```

3개의 **서로 다른** 정답을 이용해 **5초내에** 패치를 생성

Overview of CAFE

- **함수 단위**의 **context-aware matching**을 활용하여 피드백을 생성하는 기술



How CAFE Works: Running Example

- 입력 연산자에 따라 주어진 리스트의 값을 1씩 증가 혹은 감소시키는 함수

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

- I/O examples:
  - apply ([1; 2; 3], ADD) => [2; 3; 4]
  - apply ([1; 2; 3], SUB) => [0; 1; 2]

How CAFE Works: Running Example

- 입력 연산자에 따라 주어진 리스트의 값을 1씩 증가 혹은 감소시키는 함수

```
1  let rec inc_all l =
2    match l with
3    | [] -> []
4    | hd::tl -> (hd+1)::(inc_all tl)
5  let rec dec_all l =
6    match l with
7    | [] -> []
8    | hd::tl -> (hd+1)::(dec_all tl)
9  let apply (l, o) =
10   match l with
11   | [] -> []
12   | hd::tl -> (match o with
13     | ADD -> (hd+1)::(inc_all tl)
14     | SUB -> (dec_all tl))
```

- I/O examples:
  - apply ([1; 2; 3], ADD) => [2; 3; 4]
  - **apply ([1; 2; 3], SUB) => [1; 3; 4]**

- 입력 연산자가 “SUB” 일 때 오류 발생

How CAFE Works: Running Example

- 덧셈 함수, 뺄셈 함수를 각각 하나씩만 사용하여 구현된 두개의 정답 코드

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

"dec_all" 함수를 수정하기 위해, "sub_list" 함수를 사용해야 함

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

Syntax 혹은 semantic에 기반한 매칭은 잘못된 함수를 매칭

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$Ctx_{dec\_all} = (l = hd::tl) \wedge o = SUB$$

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$Ctx_{dec\_all} = (l = hd::tl) \wedge o = SUB$$

$$Ctx_{sub\_list} = (l = hd::tl) \wedge o = SUB$$

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$Ctx_{dec\_all} = (l = hd::tl) \wedge o = SUB$

$Ctx_{sub\_list} = (l = hd::tl) \wedge o = SUB$

$Ctx_{dec_all} \wedge Ctx_{sub_list}$이 **SAT**,
"dec_all" 과 "sub_list"를 매칭

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$Ctx_{dec\_all} = (l = hd::tl) \wedge o = SUB$$

$$Ctx_{add\_list} = (l = hd::tl) \wedge o = ADD$$

Step1: Context-Aware Matching

- 각 함수의 **context**에 기반하여 **패치에 유용하게 사용 될 것 같은** 정답 함수를 탐색

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$Ctx_{dec\_all} = (l = hd::tl) \wedge o = SUB$

$Ctx_{add\_list} = (l = hd::tl) \wedge o = ADD$

$Ctx_{dec\_all} \wedge Ctx_{add\_list}$ 이 **UNSAT**,
"dec_all" 과 "add_list"는 매칭되지 않음.

Step2: Template Extraction

- 서로 매칭된 함수 사이의 **syntactic difference**를 통해 패치 템플릿 추출

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

Context-aware matching에서 다음의 결과가 나왔다고 가정:
[apply ⟼ apply2, dec_all ⟼ sub_list]

Step2: Template Extraction

- 서로 매칭된 함수 사이의 **syntactic difference**를 통해 패치 템플릿 추출

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$T = \{\text{Modify}\ (8,\ \ hd + 1\ \rightarrow h - 1)\}$$

Step2: Template Extraction

- 서로 매칭된 함수 사이의 **syntactic difference**를 통해 패치 템플릿 추출

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$T = \left\{ \begin{array}{c} \text{Modify}\,(8,\ \ hd+1\ \rightarrow h-1) \\ \text{Modify}(14, \text{dec\_all}\ \ tl\ \rightarrow (h-1) :: (apply2(t,o))) \end{array} \right\}$$

Step3: Patch Generation

- 추출한 템플릿을 원본에서 사용 가능하게 수정 후, 명세를 만족가능하게 하는 **최소 패치**를 찾음

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$T = \left\{ \begin{array}{c} \text{Modify}\,(8,\ \ hd + 1 \ \rightarrow h - 1) \\ \text{Modify}(14, \text{dec\_all}\ \ tl \ \rightarrow (h-1) :: (apply2(t,o))) \end{array} \right\}$$

Step3: Patch Generation

- 추출한 템플릿을 원본에서 사용 가능하게 수정 후, 명세를 만족가능하게 하는 **최소 패치**를 찾음

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$T = \left\{ \begin{array}{c} \text{Modify}\ (8,\ \ hd + 1\ \rightarrow hd - 1) \\ \text{Modify}(14, \text{d}ec\_all\ \ tl\ \rightarrow (hd - 1) :: (dec\_all\ \ tl)) \\ \text{Modify}(14, \text{d}ec\_all\ \ tl\ \rightarrow (hd - 1) :: (dec\_all\ \ l)) \\ \text{Modify}(14, \text{d}ec\_all\ \ tl\ \rightarrow (hd - 1) :: (inc\_all\ \ tl)) \\ \text{Modify}(14, \text{d}ec\_all\ \ tl\ \rightarrow (hd - 1) :: (inc\_all\ \ l)) \end{array} \right\}$$

Step3: Patch Generation

- 추출한 템플릿을 원본에서 사용 가능하게 수정 후, 명세를 만족가능하게 하는 **최소 패치**를 찾음

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5 let rec dec_all l =
6   match l with
7   | [] -> []
8   | hd::tl -> (hd+1)::(dec_all tl)
9 let apply (l, o) =
10  match l with
11  | [] -> []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(inc_all tl)
14    | SUB -> (dec_all tl))
```

```
1 let id x = x
2
3
4
5 let rec sub_list lst =
6    match lst with
7    | [] -> []
8    | h::t -> (h-1)::(sub_list t)
9 let rec apply1 (l, o) =
10  match l with
11  | [] -> id []
12  | hd::tl -> (match o with
13    | ADD -> (hd+1)::(apply1(tl,o))
14    | SUB -> (hd-1)::(sub_list tl))
```

```
1 let rec add_list lst =
2   match lst with
3   | [] -> []
4   | h::t -> (h+1)::(add_list t)
5
6
7
8
9 let rec apply2 (l, o) =
10  match l with
11  | [] -> []
12  | h::t -> (match o with
13    | ADD -> (h+1)::(inc_all t)
14    | SUB -> (h-1)::(apply2(t,o)))
```

$$T = \left\{ \begin{array}{c} \text{Modify}\,(8,\ \boldsymbol{hd+1 \rightarrow hd-1}) \\ \text{Modify}(14,\boldsymbol{dec\_all\ tl \rightarrow (hd-1) :: (dec\_all\ tl)}) \\ \text{Modify}(14,dec\_all\ tl \rightarrow (hd-1) :: (dec\_all\ l)) \\ \text{Modify}(14,dec\_all\ tl \rightarrow (hd-1) :: (inc\_all\ l)) \end{array} \right\}$$

두 템플릿을 사용하면 최소 수정으로 패치 생성 가능

Evaluation: Performance of CAFE

- 벤치마크: 실제 학생들이 수업에서 제출한 **4,211 (3,547 correct / 664 incorrect)** 개의 OCaml 코드 사용

- 비교 대상:
  - FixML (OOPSLA 18): OCaml을 대상으로 하는 가장 최신의 피드백 생성 기술
  - SARFGEN (PLDI 18): Python을 대상으로 하는 syntax-based data-driven 피드백 생성 기술

- SARFGEN이 Python을 위한 기술이기 때문에, 다음의 두가지 버전의 SARFGEN을 구현해 실험:
  - Prog: SARFGEN의 매칭 기술을 **프로그램 단위**로 적용한 피드백 생성 기술
  - Func: SARFGEN의 매칭 기술을 **함수 단위**로 적용한 피드백 생성 기술

Evaluation: Performance of CAFE

- 다른 모든 기술에 비해 CAFE의 성능이 뛰어남을 확인
- 문제의 난이도가 어려워져도, 여전히 피드백을 잘 생성함

No	#Wrong	#Correct	FixML		Prog		Func		Cafe	
			Time	#Fix (Rate)	Time	#Fix (Rate)	Time	#Fix (Rate)	Time	#Fix (Rate)
1	45	171	0.3	40 (89%)	0.0	33 (73%)	0.0	45 (100%)	0.0	45 (100%)
2	19	117	3.1	12 (63%)	0.0	19 (100%)	0.0	19 (100%)	0.0	19 (100%)
3	9	88	0.1	7 (78%)	0.0	8 (89%)	0.0	8 (89%)	0.0	9 (100%)
4	32	704	1.6	12 (38%)	3.2	17 (53%)	3.2	17 (53%)	2.1	29 (91%)
5	49	454	11.8	26 (53%)	2.7	31 (63%)	2.6	35 (71%)	1.1	42 (86%)
6	32	125	4.1	10 (31%)	1.0	8 (25%)	1.6	9 (28%)	3.0	23 (72%)
7	35	412	23.3	18 (51%)	8.7	25 (71%)	3.1	23 (66%)	1.0	30 (86%)
8	111	597	1.5	44 (40%)	2.1	67 (60%)	1.8	71 (64%)	0.5	78 (70%)
9	141	661	1.5	23 (16%)	0.5	71 (50%)	2.7	99 (70%)	1.8	133 (94%)
10	191	218	1.1	42 (22%)	1.6	114 (60%)	2.4	118 (62%)	3.0	140 (73%)
	664	3,547	3.9	234 (35%)	1.9	393 (59%)	2.1	444 (67%)	**1.6**	**548 (83%)**

Evaluation: Performance of CAFE

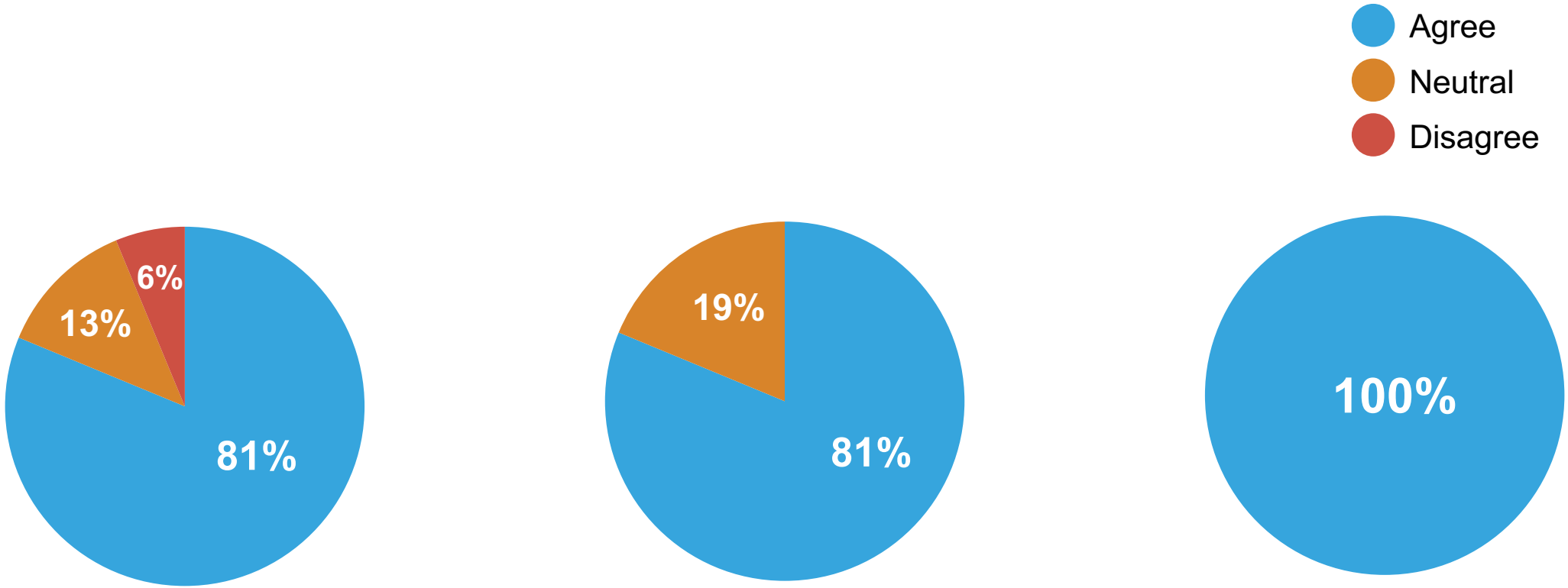
- 다른 모든 기술에 비해 CAFE의 성능이 뛰어남을 확인
- 문제의 난이도가 어려워져도, 여전히 피드백을 잘 생성함

No	#Wrong	#Correct	FixML		Prog		Func		CAFE	
			Time	#Fix (Rate)	Time	#Fix (Rate)	Time	#Fix (Rate)	Time	#Fix (Rate)
1	45	171	0.3	40 (89%)	0.0	33 (73%)	0.0	45 (100%)	0.0	45 (100%)
2	19	117	3.1	12 (63%)	0.0	19 (100%)	0.0	19 (100%)	0.0	19 (100%)
3	9	88	0.1	7 (78%)	0.0	8 (89%)	0.0	8 (89%)	0.0	9 (100%)
4	32	704	1.6	12 (38%)	3.2	17 (53%)	3.2	17 (53%)	2.1	29 (91%)
5	49	454	11.8	26 (53%)	2.7	31 (63%)	2.6	35 (71%)	1.1	42 (86%)
6	32	125	4.1	10 (31%)	1.0	8 (25%)	1.6	9 (28%)	3.0	23 (72%)
7	35	412	23.3	18 (51%)	8.7	25 (71%)	3.1	23 (66%)	1.0	30 (86%)
8	111	597	1.5	44 (40%)	2.1	67 (60%)	1.8	71 (64%)	0.5	78 (70%)
9	141	661	1.5	23 (16%)	0.5	71 (50%)	2.7	99 (70%)	1.8	133 (94%)
10										0 (73%)
										3 (83%)

중간 난이도 (82%), 고급 난이도 (79%) 문제도 높은 수치의 피드백 생성

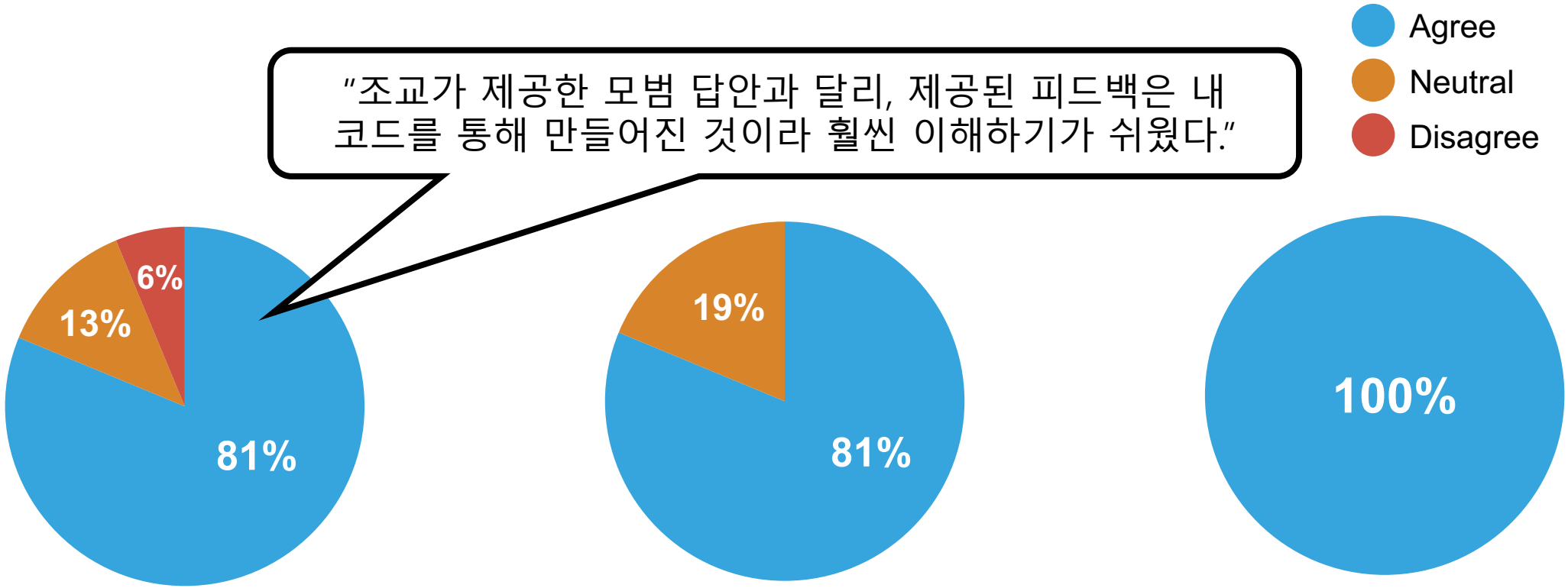
Evaluation: User Study

- 16명의 학부생들을 대상으로 다음의 설문에 대한 응답을 기록:
  - Q1: 제공된 피드백이 올바르며 이해하기 쉬운지?
  - Q2: 제공된 피드백을 통해 자신의 실수를 파악할 수 있었는지?
  - Q3: CAFE가 실제 수업에서 활용이 된다면 유용할 것 같은지?



Evaluation: User Study

- 16명의 학부생들을 대상으로 다음의 설문에 대한 응답을 기록:
  - Q1: 제공된 피드백이 올바르며 이해하기 쉬운지?
  - Q2: 제공된 피드백을 통해 자신의 실수를 파악할 수 있었는지?
  - Q3: CAFE가 실제 수업에서 활용이 된다면 유용할 것 같은지?



Discussion: Limitation of CAFE

- 피드백 생성 실패 가장 주요한 원인: 패치 생성 시간 초과
  - 현재 알고리즘은 가장 적은 수의 템플릿부터 적용을 시도
  - 작은 템플릿을 여러개 적용해야 하는 패치의 경우 패치를 잘 만들지 못함

```
1 let rec eval_exp e =
2   match e with
3   | Int n -> -> n
4   | Add (e1, e2) -> (eval_exp e1) + (eval_exp e2)
5   | Sub (e1, e2) -> (eval_exp e1) - (eval_exp e2)
6 let rec eval f =
7   match f with
8   | True -> true
9   | False -> false
10  | Not f -> if f = True then false else true
11  | AndAlso (f1, f2) -> if f1 = True && f2 = True then true else false
12  | OrElse (f1, f2) -> if f1 = False && f2 = False then false else true
13  | Imply (f1, f2) -> if f1 = True && f2 = False then false else true
14  | Equal (e1, e2) -> (eval_exp e1) = (eval_exp e2)
```

Discussion: Limitation of CAFE

- 피드백 생성 실패 가장 주요한 원인: 패치 생성 시간 초과
  - 현재 알고리즘은 가장 적은 수의 템플릿부터 적용을 시도
  - 작은 템플릿을 여러개 적용해야 하는 패치의 경우 패치를 잘 만들지 못함

```
1 let rec eval_exp e =
2   match e with
3   | Int n -> -> n
4   | Add (e1, e2) -> (eval_exp e1) + (eval_exp e2)
5   | Sub (e1, e2) -> (eval_exp e1) - (eval_exp e2)
6 let rec eval f =
7   match f with
8   | True -> true
9   | False -> false
10  | Not f -> if f = True then false else true
11  | AndAlso (f1, f2) -> if f1 = True && f2 = True then true else false
12  | OrElse (f1, f2) -> if f1 = False && f2 = False then false else true
13  | Imply (f1, f2) -> if f1 = True && f2 = False then false else true
14  | Equal (e1, e2) -> (eval_exp e1) = (eval_exp e2)
```

각 branch condition을 recursive하게 수정하면 패치 성공

Conclusion

- 학생 개개인에게 피드백을 주는 과정은 조교에게 **매우 고통스러움**
- 이 문제를 해결하기 위해 **함수 단위의 context-aware matching**을 이용한 데이터 기반 피드백 생성 기술 제안
- 실제 학생들이 제출한 OCaml에 대해서 성공적으로 피드백을 생성, 학생들에게도 도움이 됨을 확인함
- Tool: https://github.com/kupl/LearnML

Conclusion

- 학생 개개인에게 피드백을 주는 과정은 조교에게 **매우 고통스러움**
- 이 문제를 해결하기 위해 **함수 단위의 context-aware matching**을 이용한 데이터 기반 피드백 생성 기술 제안
- 실제 학생들이 제출한 OCaml에 대해서 성공적으로 피드백을 생성, 학생들에게도 도움이 됨을 확인함
- Tool: https://github.com/kupl/LearnML

감사합니다!