



# 자연어와 예제로부터 재귀함수 유도 합성하기

**Guided Recursive Program Synthesis via Natural Language and Examples**

한양대학교 프로그래밍시스템 연구실

조한결

# 합성 방식의 장단점 Neural vs Basic Search

- LLM(Large Language Models) 기반 프로그램 합성
  - High Scalability / Correctness Guarantee X
- 탐색 기반 프로그램 합성
  - Low Scalability / Correctness Guarantee O
- LLM 기반 합성의 Scalability + 탐색 기반 합성의 Correctness Guarantee

# Guided Synthesis using ChatGPT solution

- 목표 : 두 합성 방법의 장점 결합하기
  - LLM 기반 합성의 Scalability + 탐색 기반 합성의 Correctness Guarantee
  - LLM 솔루션을 탐색 기반 합성 가이드에 사용
- ChatGPT가 만드는 솔루션
  - 함수형 프로그램 합성을 위해 ChatGPT에 자연어 기술 + 입출력 예제를 제공 시 올바르지 못한 솔루션이 종종 도출됨
  - 비록 올바르게 못한 솔루션이나 정답 프로그램과 유사한 구조를 갖춘 경우가 빈번
  - ChatGPT가 생성한 프로그램과 비슷한 솔루션 우선 탐색하기

# ChatGPT의 오답 예 - I

- Q : Please give me an OCaml function that returns the n-th element of list

Example :

$f [1;2;3;4;5] 1 = 2,$

$f [2;3;4;1] 3 = 1$

- A :

```
let rec f l n =  
  match l with  
  | [] -> failwith "Empty list"  
  | x::xs -> if n = 1 then x  
              else f xs (n-1);;
```

Correct solution

**n = 0**

# ChatGPT의 오답 예 - 2

- Q : Give me an OCaml function that appends the first list at the end of the second list while removing duplicated elements

Example :

$f [1;2;3;4] [3;4;5;6;7] = [3;4;5;6;7;1;2]$ ,

$f [4;5;6;7] [1;2;3;4] = [1;2;3;4;5;6;7]$

- A :

```
let rec f list1 list2 =
```

```
  match list1 with
```

```
  | [] -> list2
```

```
  | hd::tl ->
```

```
    if List.mem hd list2 then f tl list2
```

```
    else f tl (hd::list2)
```

(list2 @ [hd])

Correct solution

# ChatGPT의 오답 예 - 3

- Q : Give me an OCaml function that returns a list comprising only the second element of each list in a given list.

Example:

`f [[1;2;3] [4;2;3] [2;2;2;3]] -> [2;2;2],`

`f [[1] [2] [3;4] ]-> [4],`

`f [[0;1]] -> [1]`

- A : 

```
let rec f lst =  
  match lst with  
  | [] -> []  
  | x :: xs -> (match x with  
    | y :: ys -> y :: (f xs) Correct solution  
    | _ -> f xs)
```

`_ :: y :: ys`

# ChatGPT의 오답 예 - 4

- Q : give me ocaml function that takes the last n elements of a list l.

$f [1;2;3;4;5] 3 = [3;4;5],$

$f [1;2;3] 1 = [3]$

$f [3;2;1] 2 = [2;1]$

$l\ n$

Correct solution

- A : 

```
let rec f n l =  
  match l with  
  | [] -> []  
  | hd :: tl ->  
    if n <= 0 then l  
    else f (n - 1) tl
```

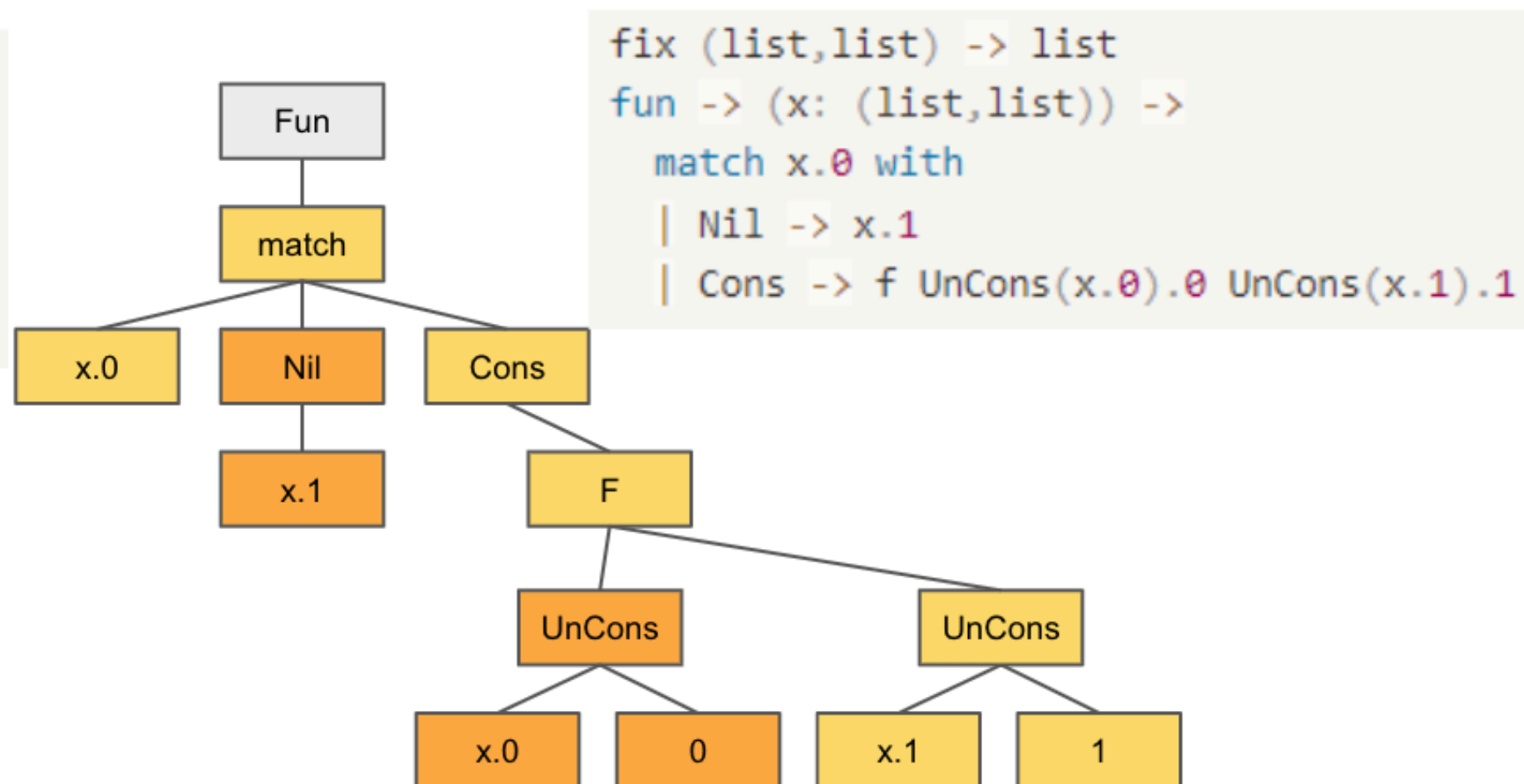
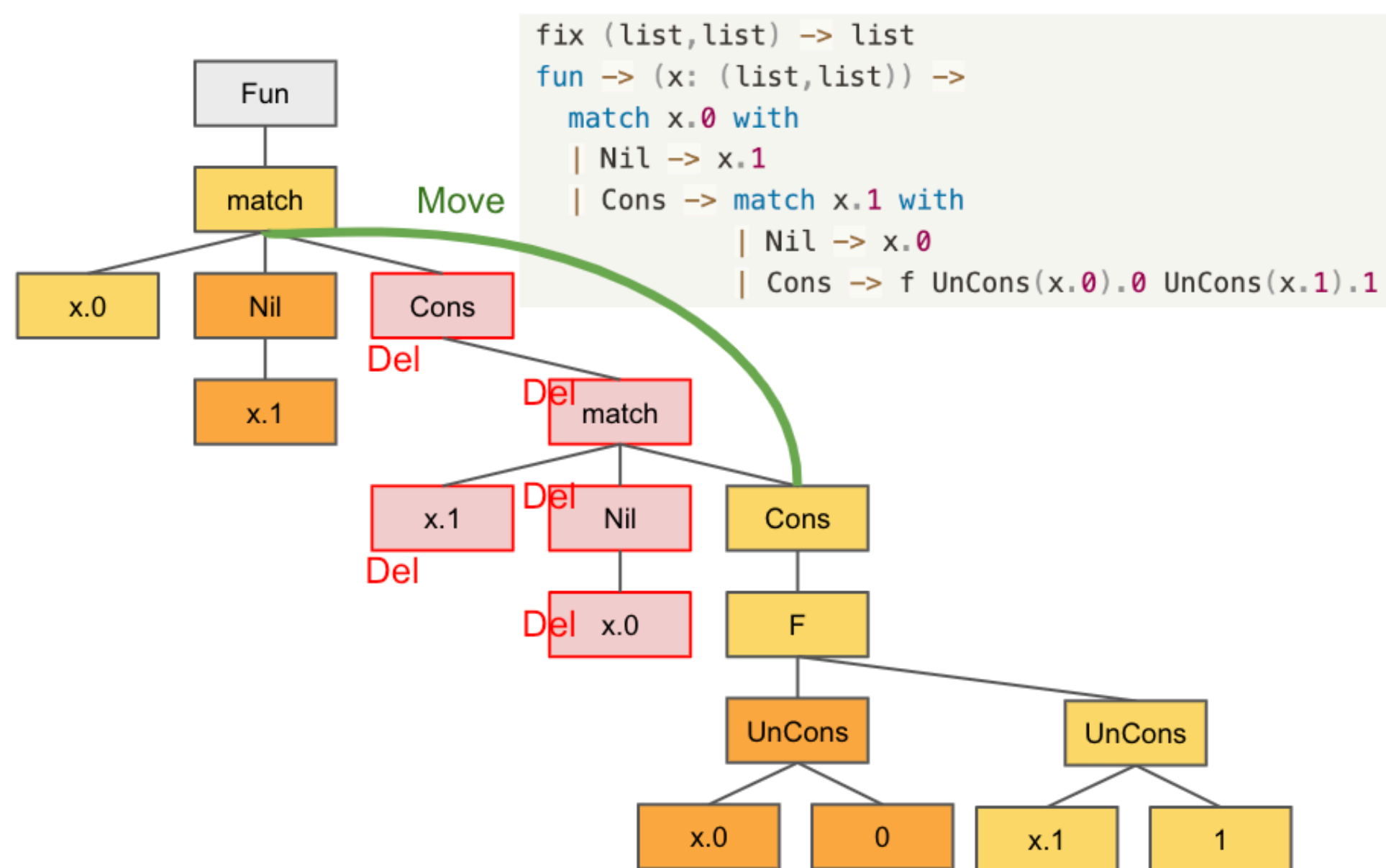
```
if n = 0 then []  
else if List.length tl < n  
then l  
else f tl n
```

# 방법

- 탐색 시 ChatGPT의 솔루션과 유사한 것 먼저 고려하기

- 유사 측정: Tree edit distance 사용

Tree edit distance: 원본 트리에서 목적 트리로 (1)노드 삽입, (2)노드 삭제, (3)서브트리 이동, (4)노드 레이블 바꾸기 네 가지 명령을 통해 변경하기 위한 최단 명령 수



노드 삭제  
노드 삭제  
노드 삭제  
노드 삭제  
노드 삭제  
서브트리 이동

---

Distance = 6



# 결과 : 합성 가속화 가능성 확인

- Q : give me ocaml function that takes the last n elements of a list l.

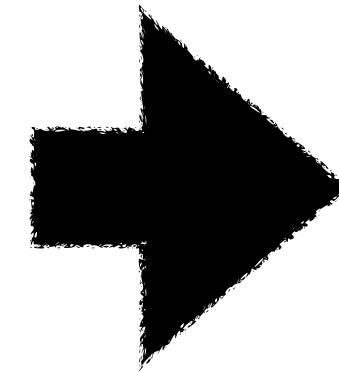
f [1;2;3;4;5] 3 = [3;4;5],

f [1;2;3] 1 = [3]

f [3;2;1] 2 = [2;1]

- ChatGPT Answer :

```
let rec f n l =  
  match l with  
  | [] -> []  
  | hd :: tl ->  
    if n <= 0 then l  
    else f (n - 1) tl
```



Non guided Trio

Timeout 10m

Guided Trio

> 1s

```
let rec (f : ((list, nat) -> list)) =  
  fun (x:(list, nat)) ->  
    match (x).0 with  
    Nil(_) ->  
      (x).0  
    Cons(_) ->  
      match ((sub (length (x).0)) S((x).1)) with  
      0(_) ->  
        (Un_Cons((x).0)).1  
      S(_) ->  
        match ((sub (x).1) (Un_Cons((x).0)).0) with  
        0(_) ->  
          Cons([(length (x).0), Nil([])])  
        S(_) ->  
          (f [(Un_Cons((x).0)).1, (length (x).0)])
```

# 기대효과

- ChatGPT의 부분적으로 올바른 코드를 탐색 가이드에 활용하여 **합성 가속화**
- ChatGPT가 보장하지 못하는 입출력 예제에 대한 **올바름 보장**
- 예제는 만족하지만 기대와는 다른 **과적합 문제 회피**
- 자세한 부분은 Poster Section에서..