프로그램 합성

이 우 석

한양대학교 ERICA 소프트웨어학부

2021. 03. 19 @ 충남대학교

내용

- **프로그램 합성 개괄**
- 귀납 합성 전략
- 연역 합성 전략

프로그램 합성

- 사용자가 **원하는** 프로그램을 **자동**으로 생성하는 기술

프로그래밍 할 줄 몰라도
표현할 수 있어야
(예: 입출력 예제)

- 1945년부터 언급(앨런 튜링), 실현될 경우 개발의 무거운 짐을 덜어줄 것으로 기대되는 컴퓨터공학의 성배(holy grail)

- 1960년대부터 느리게 연구되어오다가 2008년 CEGIS라는 기술의 등장을 기점으로 최근까지 두드러지게 발전

프로그램 합성과 연관기술 비교 (합성 vs. 컴파일러)

```
let rec insert x xs =
    match xs with
    | [] -> x :: []
    | h :: t ->
if x <= h then x :: xs
else h :: (insert x t)
```

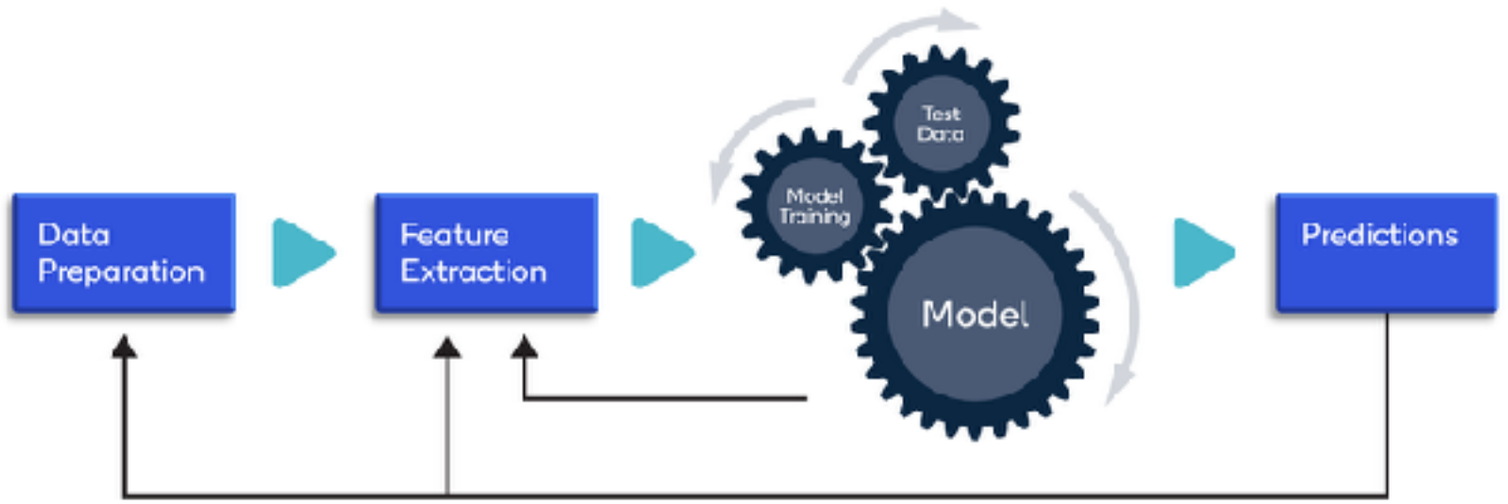
OCaml

```
append:
    push ebp
    mov ebp, esp
    push eax
    push ebx
    push len
    call malloc
    mov ebx, [ebp + 12]
    mov [eax + info], ebx
    mov dword [eax + next], 0
    mov ebx, [ebp + 8]
    cmp dword [ebx], 0
    je null_pointer
    mov ebx, [ebx]

next_element:
    cmp dword [ebx + next], 0
    je found_last
    mov ebx, [ebx + next]
    jmp next_element

found_last:
    push eax
    push addMes
    call puts
    add esp, 4
    pop eax
    mov [ebx + next], eax

go_out:
    pop ebx
    pop eax
    mov esp, ebp
    pop ebp
    ret 8

null_pointer:
    push eax
    push nullMes
    call puts
    add esp, 4
    pop eax
    mov [ebx], eax
    jmp go_out
```

어셈블리

- **공통점**: 고 수준 (high level, 사람이 이해할 수 있는) 언어로 쓰인 개념을 실행 가능한 프로그램으로 변환
- **차이점**:
  - 컴파일러: 탐색 과정 없이 단순 변환
  - 합성:“어떻게” 주어진 목표를 달성할지 컴퓨터가 **탐색**

프로그램 합성과 연관기술 비교
(합성 vs. 기계학습)



- **공통점**: 주어진 입출력 데이터에 맞게 작동하는 함수 찾기

- **차이점**:

  - 기계학습: 튜링완전turing complete하지 않은 대상(결정트리, 인공신경망 구조 등) 학습, (대개)결과물 **해독 불가**, 주어진 데이터에 꼭 맞지 않아도 됨 (노이즈 허용).

  - 합성: 일반 프로그램 대상(튜링 완전), 해독 가능, 주어진 데이터에 꼭 맞아야.

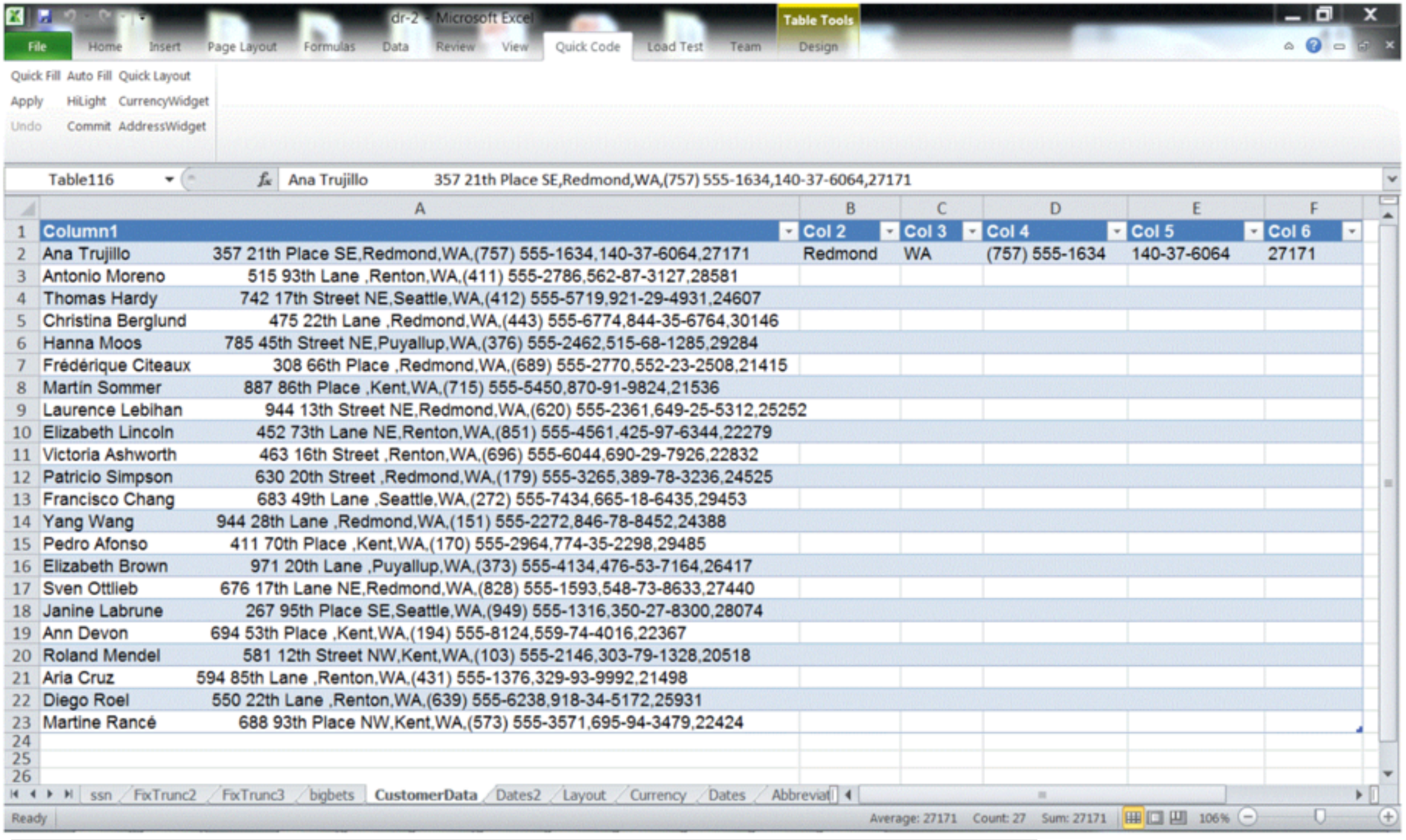
프로그램 합성과 연관기술 비교
(합성 vs. 선언형 프로그래밍)

프롤로그(Prolog)의 예:

```
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
parent(X, Y) :- father(X, Y).
mother(X, Y) :- father(X, Y).

mother(Mary, Stan). Father(Stan, Alice).
```

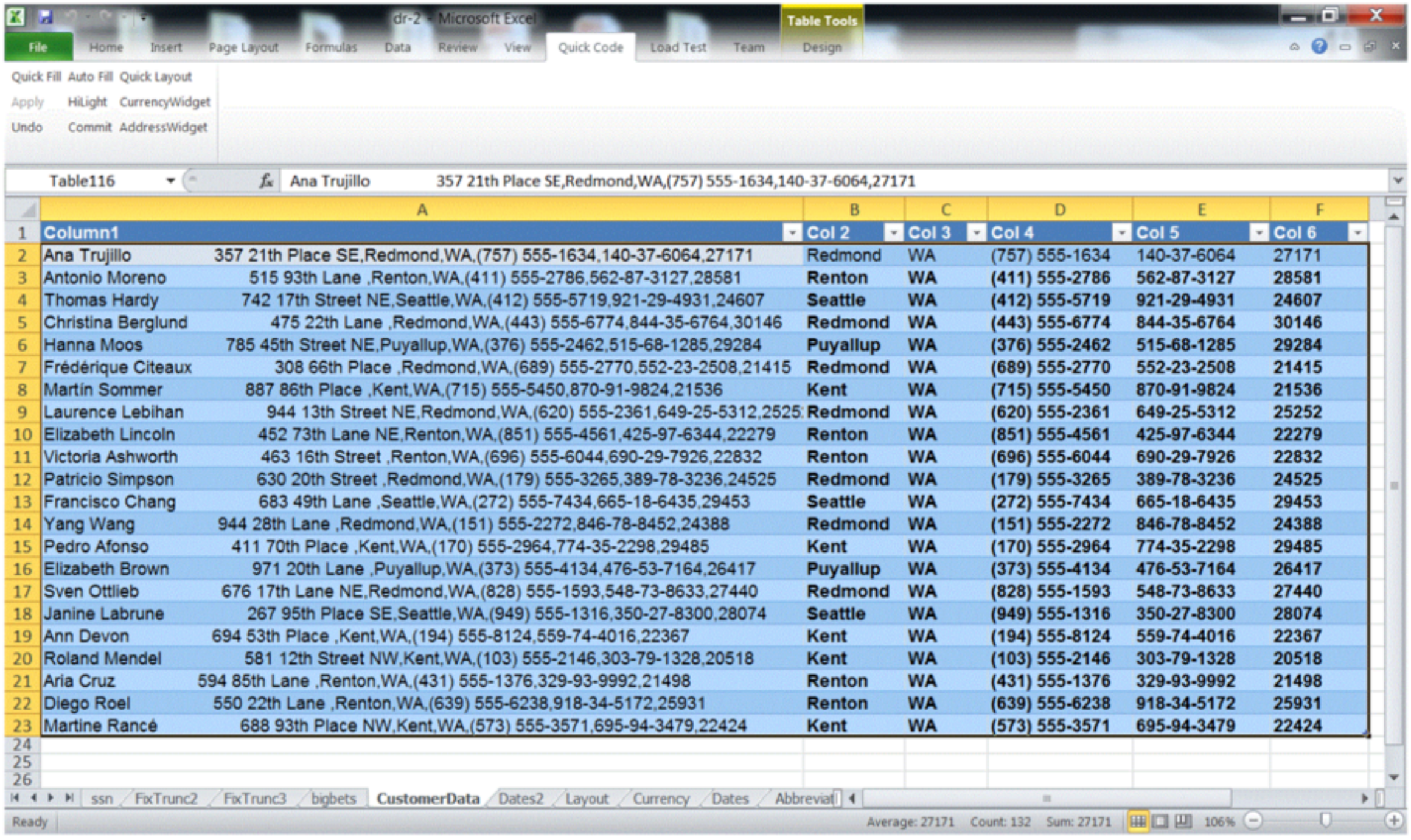
- **공통점**: 사용자는 계산과정에 대한 얘기없이 결과물에 대한 조건만 기술
- **차이점**:
  - 선언형 프로그램: 요구조건에 **모호성이 없어야** 함 (예: 입출력 예제는 안됨), 결과물의 실행 성능 느림.
  - 합성: 요구조건이 모호해도 되고 결과물은 일반 프로그램

응용사례 – 문자열 변환 (FlashFill)



Automating String Processing in Spreadsheets using Input-Output Examples, POPL'11

응용사례 – 문자열 변환 (FlashFill)

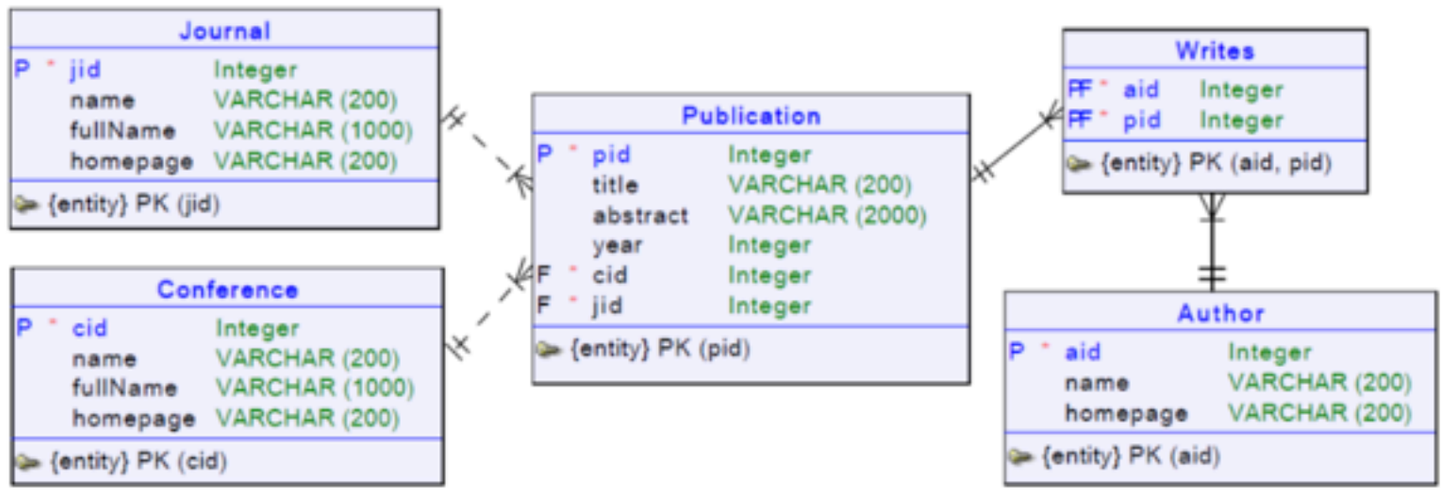


	A	B	C	D	E	F
1	Column1	Col 2	Col 3	Col 4	Col 5	Col 6
2	Ana Trujillo 357 21th Place SE,Redmond,WA,(757) 555-1634,140-37-6064,27171	Redmond	WA	(757) 555-1634	140-37-6064	27171
3	Antonio Moreno 515 93th Lane ,Renton,WA,(411) 555-2786,562-87-3127,28581	Renton	WA	(411) 555-2786	562-87-3127	28581
4	Thomas Hardy 742 17th Street NE,Seattle,WA,(412) 555-5719,921-29-4931,24607	Seattle	WA	(412) 555-5719	921-29-4931	24607
5	Christina Berglund 475 22th Lane ,Redmond,WA,(443) 555-6774,844-35-6764,30146	Redmond	WA	(443) 555-6774	844-35-6764	30146
6	Hanna Moos 785 45th Street NE,Puyallup,WA,(376) 555-2462,515-68-1285,29284	Puyallup	WA	(376) 555-2462	515-68-1285	29284
7	Frédérique Citeaux 308 66th Place ,Redmond,WA,(689) 555-2770,552-23-2508,21415	Redmond	WA	(689) 555-2770	552-23-2508	21415
8	Martín Sommer 887 86th Place ,Kent,WA,(715) 555-5450,870-91-9824,21536	Kent	WA	(715) 555-5450	870-91-9824	21536
9	Laurence Lebihan 944 13th Street NE,Redmond,WA,(620) 555-2361,649-25-5312,2525	Redmond	WA	(620) 555-2361	649-25-5312	25252
10	Elizabeth Lincoln 452 73th Lane NE,Renton,WA,(851) 555-4561,425-97-6344,22279	Renton	WA	(851) 555-4561	425-97-6344	22279
11	Victoria Ashworth 463 16th Street ,Renton,WA,(696) 555-6044,690-29-7926,22832	Renton	WA	(696) 555-6044	690-29-7926	22832
12	Patricio Simpson 630 20th Street ,Redmond,WA,(179) 555-3265,389-78-3236,24525	Redmond	WA	(179) 555-3265	389-78-3236	24525
13	Francisco Chang 683 49th Lane ,Seattle,WA,(272) 555-7434,665-18-6435,29453	Seattle	WA	(272) 555-7434	665-18-6435	29453
14	Yang Wang 944 28th Lane ,Redmond,WA,(151) 555-2272,846-78-8452,24388	Redmond	WA	(151) 555-2272	846-78-8452	24388
15	Pedro Afonso 411 70th Place ,Kent,WA,(170) 555-2964,774-35-2298,29485	Kent	WA	(170) 555-2964	774-35-2298	29485
16	Elizabeth Brown 971 20th Lane ,Puyallup,WA,(373) 555-4134,476-53-7164,26417	Puyallup	WA	(373) 555-4134	476-53-7164	26417
17	Sven Ottlieb 676 17th Lane NE,Redmond,WA,(828) 555-1593,548-73-8633,27440	Redmond	WA	(828) 555-1593	548-73-8633	27440
18	Janine Labrune 267 95th Place SE,Seattle,WA,(949) 555-1316,350-27-8300,28074	Seattle	WA	(949) 555-1316	350-27-8300	28074
19	Ann Devon 694 53th Place ,Kent,WA,(194) 555-8124,559-74-4016,22367	Kent	WA	(194) 555-8124	559-74-4016	22367
20	Roland Mendel 581 12th Street NW,Kent,WA,(103) 555-2146,303-79-1328,20518	Kent	WA	(103) 555-2146	303-79-1328	20518
21	Aria Cruz 594 85th Lane ,Renton,WA,(431) 555-1376,329-93-9992,21498	Renton	WA	(431) 555-1376	329-93-9992	21498
22	Diego Roel 550 22th Lane ,Renton,WA,(639) 555-6238,918-34-5172,25931	Renton	WA	(639) 555-6238	918-34-5172	25931
23	Martine Rancé 688 93th Place NW,Kent,WA,(573) 555-3571,695-94-3479,22424	Kent	WA	(573) 555-3571	695-94-3479	22424
24						
25						
26						

Automating String Processing in Spreadsheets using Input-Output Examples, POPL'11

응용사례 – SQL 합성 (SQLizer)

Problem: "Find the number of papers in OOPSLA 2010"

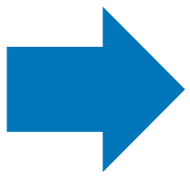


Output:

```
SELECT count(Publication.pid)
FROM Publication JOIN Conference ON Publication.cid = Conference.cid
WHERE Conference.name = "OOPSLA" AND Publication.year = 2010
```

응용사례 – 수퍼최적화 (STOKE)

```
 1 # gcc -O3
 2
 3 movq rsi, r9
 4 movl ecx, ecx
 5 shrq 32, rsi
 6 andl 0xffffffff, r9d
 7 movq rcx, rax
 8 movl edx, edx
 9 imulq r9, rax
10 imulq rdx, r9
11 imulq rsi, rdx
12 imulq rsi, rcx
13 addq rdx, rax
14 jae .L0
15 movabsq 0x100000000, rdx
16 addq rdx, rcx
17 .L0:
18 movq rax, rsi
19 movq rax, rdx
20 shrq 32, rsi
21 salq 32, rdx
22 addq rsi, rcx
23 addq r9, rdx
24 adcq 0, rcx
25 addq r8, rdx
26 adcq 0, rcx
27 addq rdi, rdx
28 adcq 0, rcx
29 movq rcx, r8
30 movq rdx, rdi
```



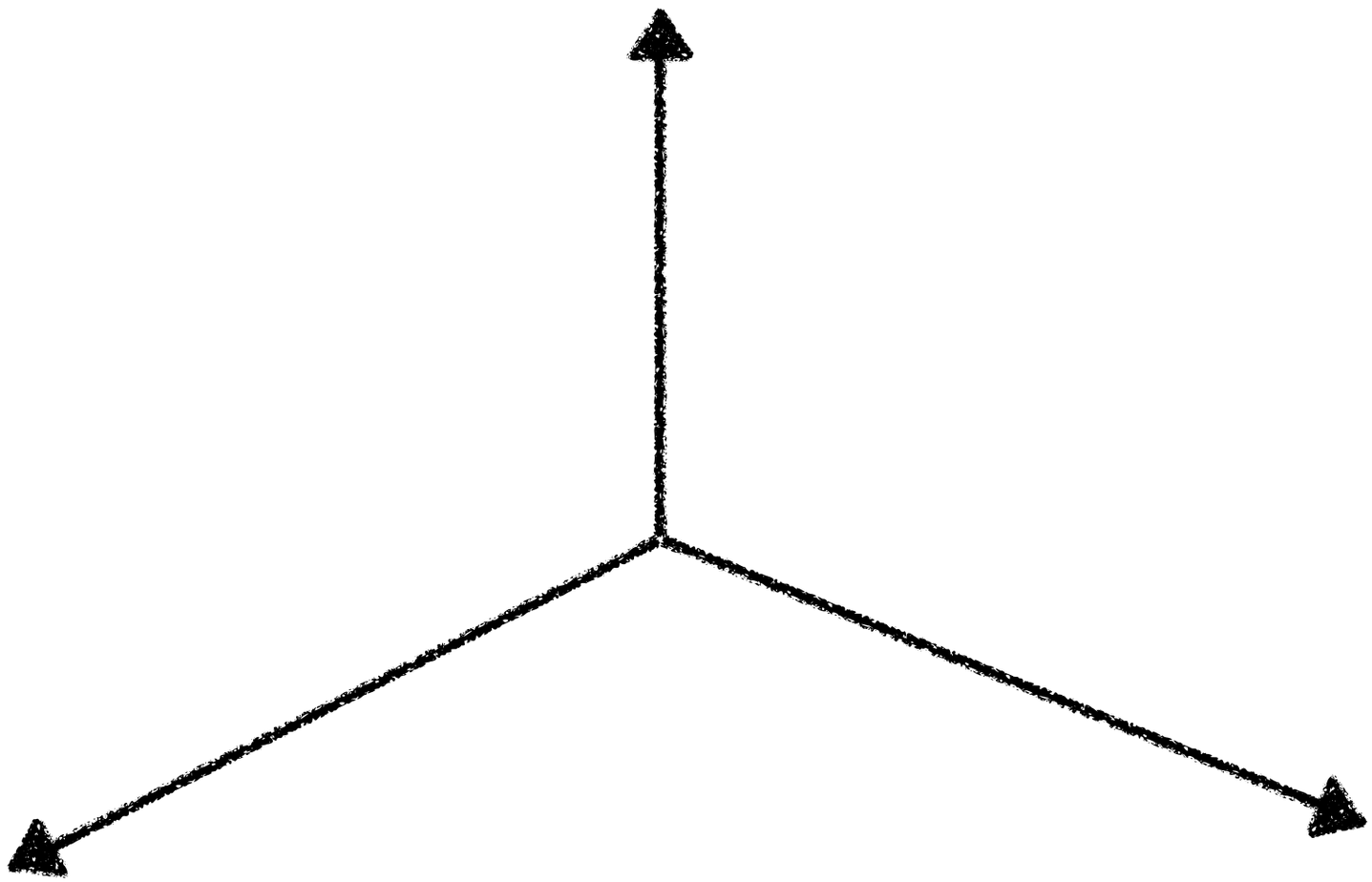
```
 1 # STOKE
 2
 3 shlq 32, rcx
 4 movl edx, edx
 5 xorq rdx, rcx
 6 movq rcx, rax
 7 mulq rsi
 8 addq r8, rdi
 9 adcq 0, rdx
10 addq rdi, rax
11 adcq 0, rdx
12 movq rdx, r8
13 movq rax, rdi
```

Montgomery multiplication kernel from the OpenSSL RSA library. Compilations shown for gcc –O3 (left) and a stochastic optimizer (right).

http://stoke.stanford.edu

프로그램 합성의 차원

제약 조건: 프로그램이 달성할 목표를 어떻게 기술할지



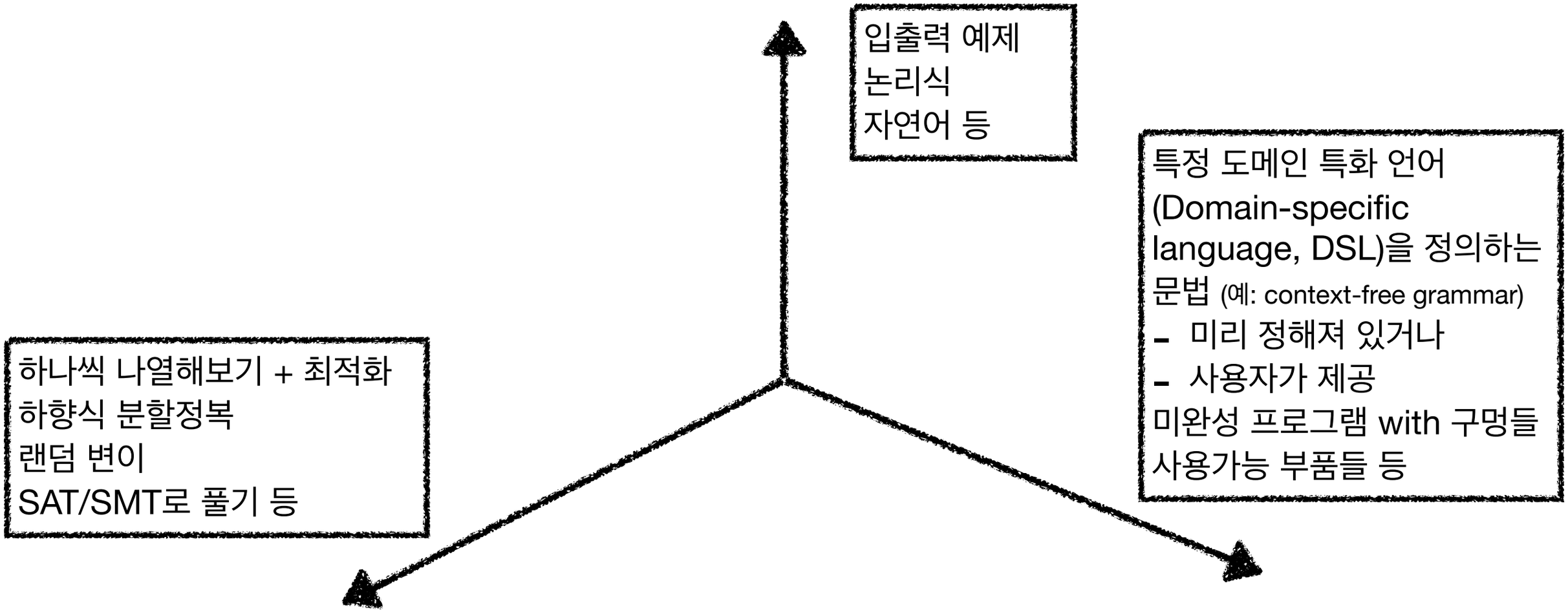
탐색 전략: 원하는 프로그램을 어떻게 찾을지

탐색 공간: 대상 프로그램 후보군은 무엇인지

Dimensions in program synthesis, PPDP’10

프로그램 합성의 차원

제약 조건: 프로그램이 달성할 목표를 어떻게 기술할지



탐색 전략: 원하는 프로그램을 어떻게 찾을지

탐색 공간: 대상 프로그램 후보군은 무엇인지

Dimensions in program synthesis, PPDP'10

FlashFill (Excel 문자열 변환)

제약 조건: 프로그램이 달성할 목표를 어떻게 기술할지

입출력 예제
논리식
자연어 등

제약조건이 입출력 예제인
경우에만 적용가능

하나씩 나열해보기 + 최적화
하향식 분할정복
랜덤 변이
SAT/SMT로 풀기 등

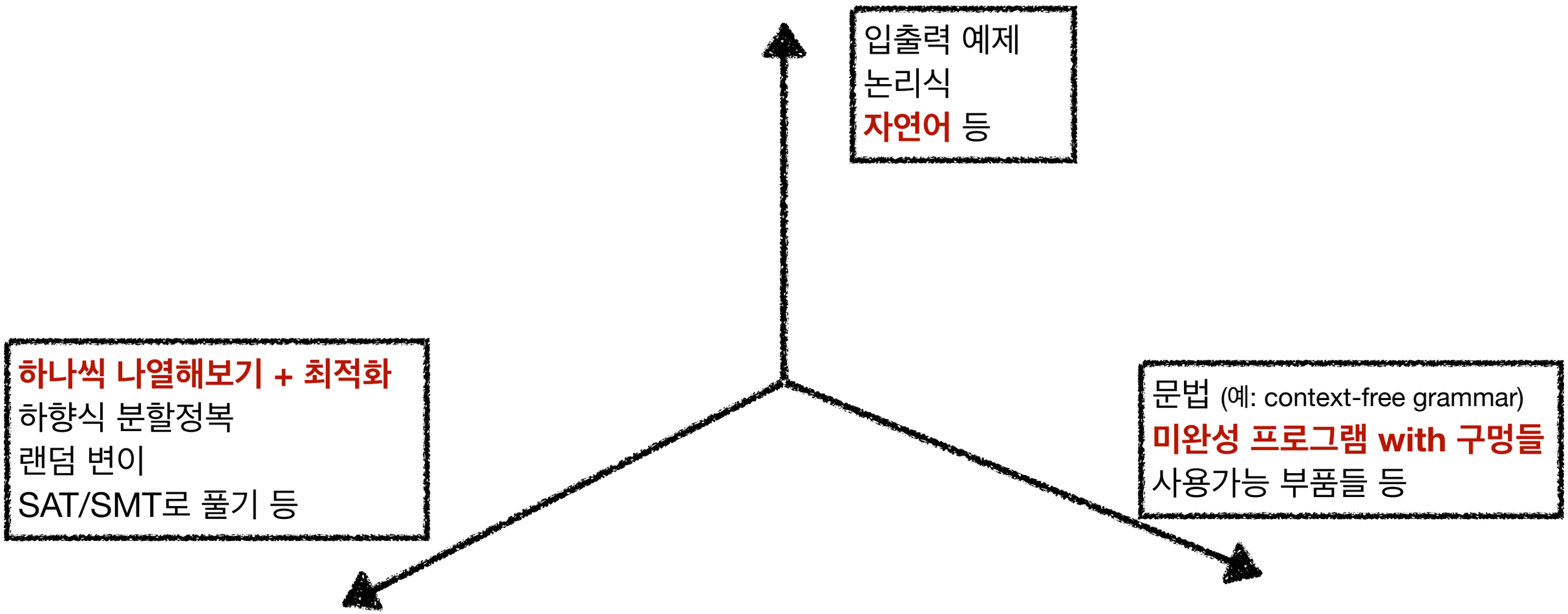
문법 (예: context-free grammar)
미완성 프로그램 with 구멍들
사용가능 부품들 등

탐색 전략: 원하는 프로그램을 어떻게 찾을지

탐색 공간: 대상 프로그램 후보군은 무엇인지

SQLizer (DB질의문 합성)

제약 조건: 프로그램이 달성할 목표를 어떻게 기술할지

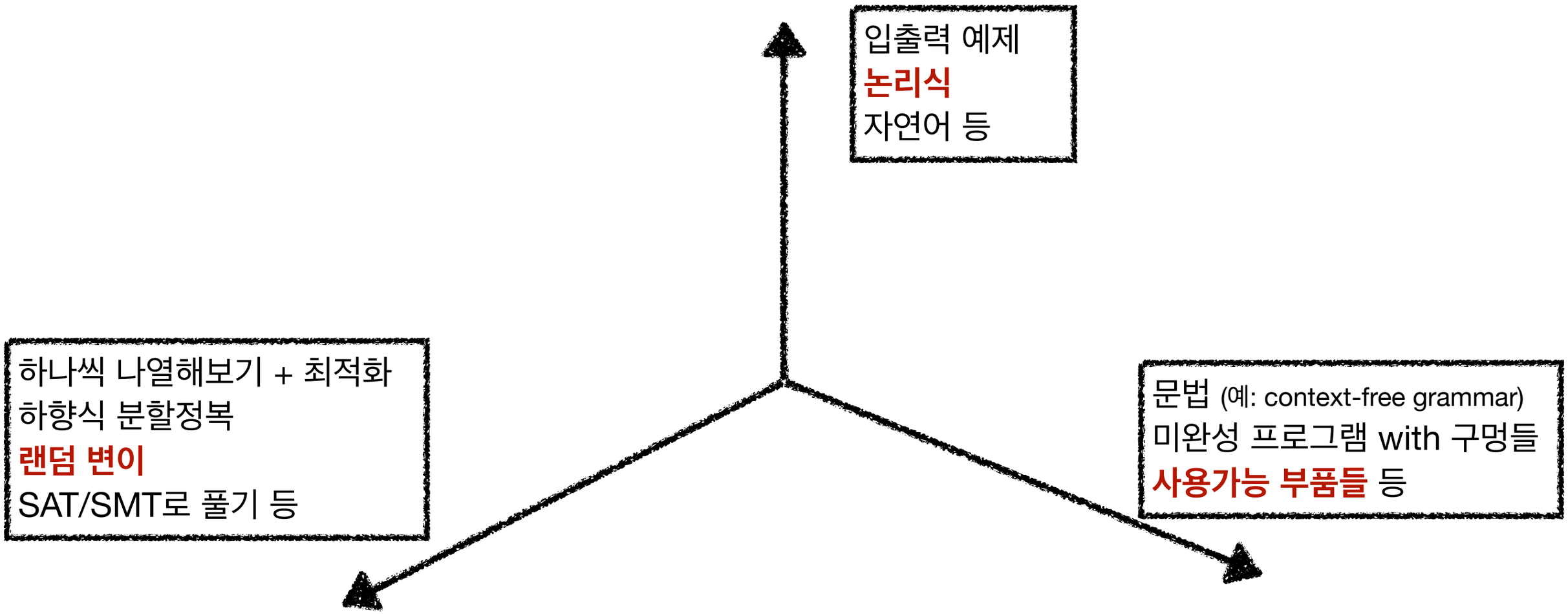


탐색 전략: 원하는 프로그램을 어떻게 찾을지

탐색 공간: 대상 프로그램 후보군은 무엇인지

STOKE (프로그램 최적화)

제약 조건: 프로그램이 달성할 목표를 어떻게 기술할지



탐색 전략: 원하는 프로그램을 어떻게 찾을지

탐색 공간: 대상 프로그램 후보군은 무엇인지

너무 다양한 합성 문제들

- 각 응용 도메인에 대해서 따로따로 개발되는 합성 기술들의 비교가 어려움.

- **관찰**:
  - 대부분의 합성 문제가 문맥 둔감 문법(context-free grammar)로 기술되는 프로그램 공간에서 하나의 해를 찾는 문제임.

  - 또한, 원하는 프로그램에 대한 제약조건이 대개의 경우 SMT 논리식으로 작성될 수 있음.

너무 다양한 합성 문제들

- **결과**:
  - 합성 문제를 기술하는 표준화된 포맷 (Syntax-guided Synthesis, SyGuS) 고안.
  - 이 포맷으로 기술된 합성 문제를 푸는데 사용될 수 있는 다목적(general-purpose) 합성 전략들 등장
  - 공통 포맷으로 인해 서로다른 합성기들 성능비교 가능해짐. 매년 합성기 경진대회 (SyGuS competition) 개최.

Syntax-Guided Program Synthesis (SyGuS)†

다양한 종류의 합성 문제들을 기술하기 위한 표준화된 포맷

생김새 조건: 문맥 둔감 문법(context-free grammar).
SMT Solver 가 이해할 수 있는 연산자들 사용 가능.
탐색 공간을 적절히 조절.

Specification

Syntactic constraint

$$S \to x \mid S \times S \mid 1 \mid 2 \mid \cdots$$

Semantic constraint

$$f(1) = 2 \wedge f(3) = 6$$

행동 조건: 합성 대상 함수 f 가 만족시켜야 하는 제약조건 (SMT 논리식)

Synthesizer

Program

$$f(x) = 2x$$

†http://www.sygus.org

합성전략의 두 갈래

- 제약조건의 기술 방법에 따라 전략이 두 갈래로 갈림.

- **귀납 합성** (inductive synthesis) — 제약조건이 입출력 예제 (a.k.a. Programming by example)
  - 합성된 프로그램의 올바름 체크: 입출력 매치

- **연역 합성** (deductive synthesis) — 제약조건이 논리식
  - 합성된 프로그램의 올바름 체크: SAT/SMT solver 사용

내용

- 프로그램 합성 개괄
- **귀납 합성 전략**
- 연역 합성 전략

귀납 합성 전략

- 하나씩 나열해보기 + 최적화 (일반적으로 적용 가능, 느림)
  - 하향식 (Top-Down) / 상향식 (Bottom-Up)
  - 최적화: 통계모델 사용 (PLDI'18), 프로그램 분석 등
- 하향식 분할정복 (특정 문제들에만 적용 가능, 빠름)
  - 제품에 탑재될 정도로 빠름 (Excel, PowerShell, Visual Studio)
- 상향식 나열 + 효율적 자료구조 쓰기 (일반적으로 적용 가능, 빠름)
  - 최근의 연구 성과 (POPL'21)

귀납 합성 전략

- 하나씩 나열해보기
  - **하향식 (Top-Down)**
  - 상향식 (Bottom-Up)
    - 하향식 + 통계모델 사용 (PLDI'18)
- 하향식 분할정복
- 상향식 나열 + 효율적 자료구조 쓰기 (POPL'21)

하향식(Top-Down) 나열

- 시작 비말단기호(start non-terminal)에서 시작
- 비말단 기호하나를 (보통 가장 왼쪽의 것) 선택하여 문법 규칙을 적용하여 확장
- 예제: 합성대상 $f : \mathbb{Z} \to \mathbb{Z}$

  생김새 조건: $S \to x \mid 1 \mid -S \mid S + S \mid S \times S$

  행동조건: $f(1) = 2$

하향식(Top-Down) 나열

Iter 0	S				
Iter 1	x	1	$-S$	$S+S$	$S \times S$
Iter 2	$-x$	-1	$-(-S)$	$x+S \cdots$	
Iter 3	$-(-x)$	$-(-1)$	$\boxed{x+1}$		

하향식 나열 최적화

- 탐색 중 미완성 프로그램 (예: x + S) 들 등장.
- 미완성 프로그램의 행동을 미리 어림잡아 (프로그램 분석) 해가 될 가능성이 없는 것들은 미리 가지치기 가능
- 예: 입출력 예제 조건이 $f(2) = 3$ 일 경우 $x \times S$ 는 후보에서 제외하고 더 탐색 진행 안함
- 타입/SMT제약식 풀기/요약해석 등 다양하게 이용.

귀납 합성 전략

- 하나씩 나열해보기
  - 하향식 (Top-Down)
  - **상향식 (Bottom-Up)**
    - 하향식 + 통계모델 사용 (PLDI'18)
- 하향식 분할정복
- 상향식 나열 + 효율적 자료구조 쓰기 (POPL'21)

상향식(Bottom-Up) 나열

- 말단기호(terminal)에서 시작
- 가장 작은 프로그램들 부터 만들고, 그것들을 조합하여 더 큰 프로그램을 만듦.

상향식(Bottom-Up) 나열

Size 1

$x \quad 1$

Size 2

$-x \quad -1$

Size 3

$1 \times 1 \quad 1 \times x \quad x \times 1 \quad x \times x$

$x + 1$

상향식 나열 최적화

- 탐색 중 완성 프로그램 (예: I + I) 들 등장. 완성 프로그램들은 실행이 가능
- 실행 결과가 같은 부품들 중 하나만 남겨놓고 더 큰 프로그램 생성에 사용.
- 이를 같은 것들 (observational equivalence) 뭉뚱그리기 라고 함.

상향식(Bottom-Up) 나열

$$x = 1 \because f(1) = 2$$

Size 1

x ~~1~~

Size 2

$-x$ ~~-1~~

Size 3

~~1×1~~ ~~$1 \times x$~~ ~~$x \times 1$~~ ~~$x \times x$~~

$x + 1$

하나씩 나열하기

- 장점: 귀납 합성 뿐 아니라 일반적으로 거의 모든 종류의 프로그램 합성 문제에 사용 가능

- 단점: 탐색 공간이 너무나 크므로, 여러가지 최적화(예: 같은 것 뭉뚱그리기)에도 불구하고 큰 프로그램 찾기 불가능

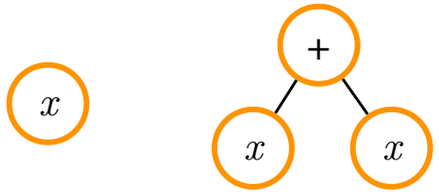
탐색공간 얼마나 큰 가?

$$S \to x \mid S + S$$

depth ≤ 1

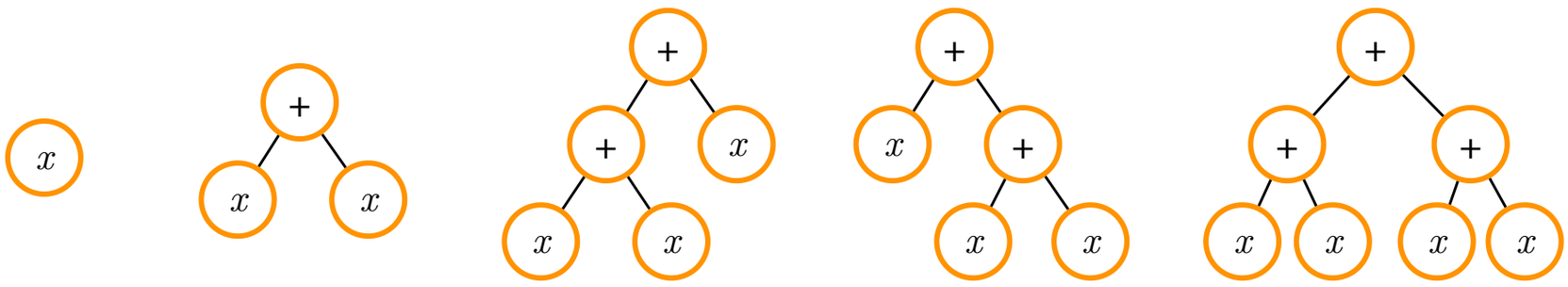
N(1) = 1

depth ≤ 2



N(2) = 2

depth ≤ 3



N(3) = 5

N(d) = 1 + N(d - 1)²

*Examples from Nadia Polikarpova's slides

탐색공간 얼마나 큰 가?

$$S \to x \mid S + S$$

N(d) = 1 + N(d - 1)²

N(1) = 1
N(2) = 2
N(3) = 5
N(4) = 26
N(5) = 677
N(6) = 458330
N(7) = 210066388901
N(8) = 44127887745906175987802
N(9) = 19472704769152964495597034454938489304527912 05
N(10) = 37918623102659260828682350280278932773702331522473885847617341507177682544103411753253520 26

!!

*Examples from Nadia Polikarpova's slides

귀납 합성 전략

- 하나씩 나열해보기
  - 하향식 (Top-Down)
  - 상향식 (Bottom-Up)
    - **하향식 + 통계모델 사용 (PLDI'18)**
- 하향식 분할정복
- 상향식 나열 + 효율적 자료구조 쓰기 (POPL'21)

통계 모델을 이용한 탐색 가이드

- 하나씩 나열하기의 성능 문제는 탐색 공간에 존재하는 모든 프로그램을 모두 "**공평**"하게 대하기 때문.

- 모든 프로그램이 "**그럴싸**"한 것은 아님. 프로그램에는 반복적이고 예측 가능한 패턴이 존재.

```
for (i = 0; i < 100; ??)
```

- 프로그램의 통계적 모델: 프로그램에 대한 확률 분포 결정

$$Pr(?? \rightarrow \texttt{i++} \mid \texttt{for(i=0;i<100;??))} = 0.85$$

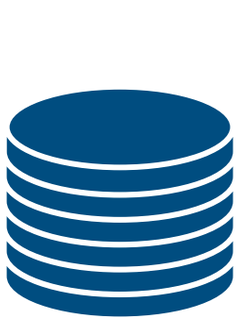
$$Pr(?? \rightarrow \texttt{i--} \mid \texttt{for(i=0;i<100;??))} = 0.01$$

통계 모델을 이용한 탐색 가이드

- 통계 프로그램 모델은 역난독화, IDE의 코드 완성, 프로그램 자동 수정 등에 이미 사용되어옴
  - N-gram, Probabilistic context-free grammar (PCFG) 등
- 아이디어: 프로그램 크기 오름차순이 아닌, 그럴싸함 확률 내림차순으로 나열

통계적 프로그램 모델

- 사람이 쓴, 혹은 합성된 코드로부터 학습 가능



Program Corpus

$Pr(S \rightarrow S + S) = 0.3$

$Pr(S \rightarrow x \mid S + S) = 0.8$

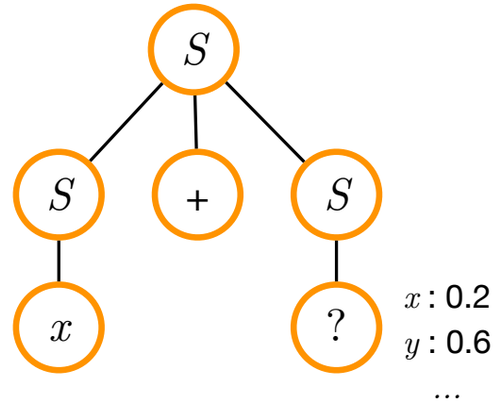
...

$Pr(S \rightarrow x \mid x + S) = 0.2$

$Pr(S \rightarrow y \mid x + S) = 0.6$

...

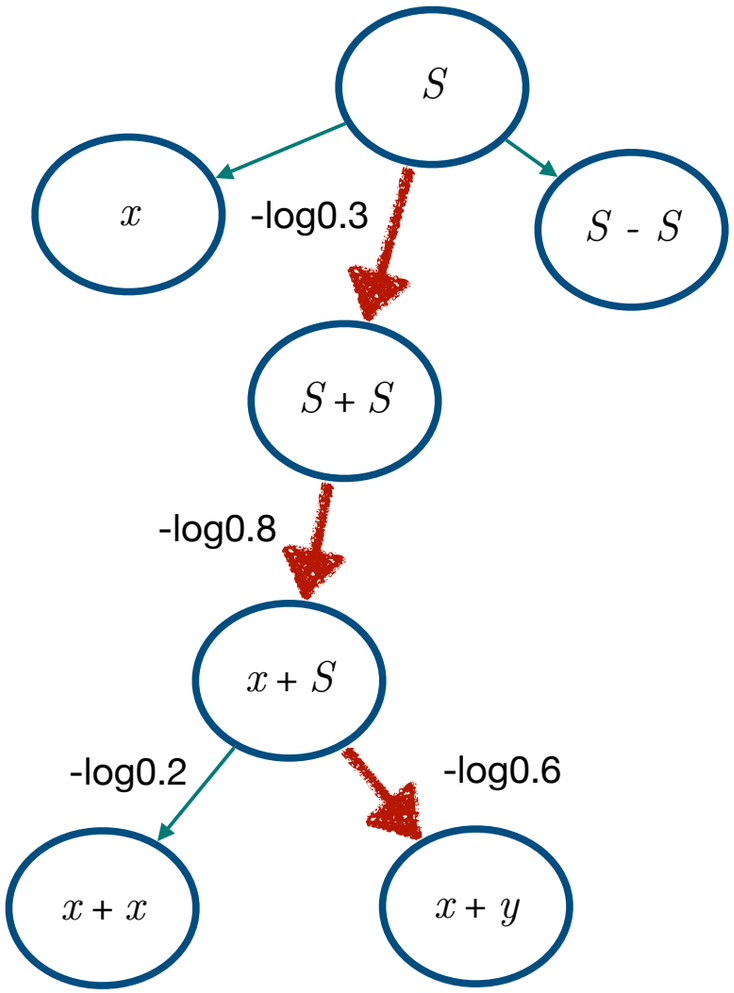
Learned Probabilistic Model



Probability of Programs

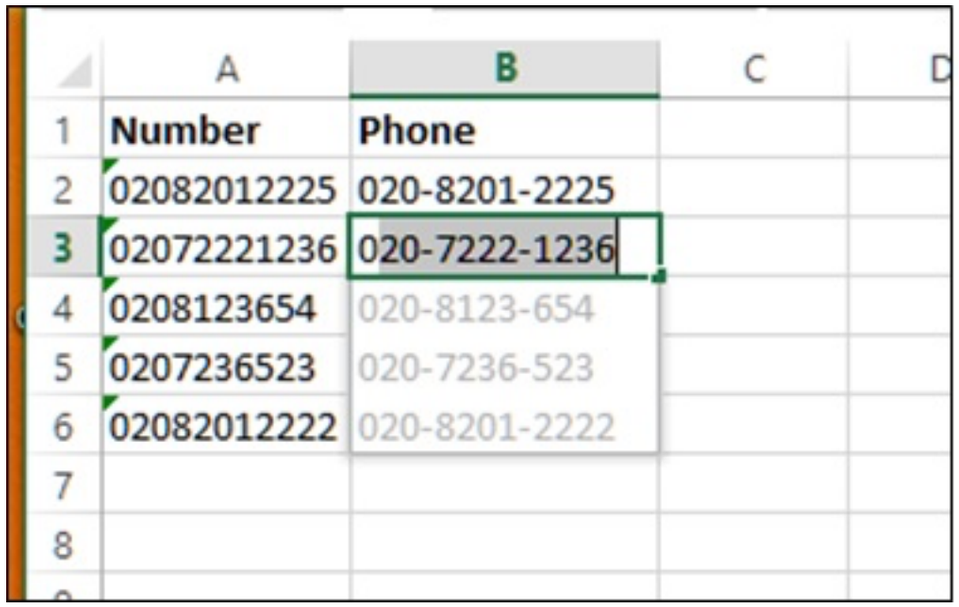
통계 모델을 이용한 탐색 가이드

- 통계 모델로부터 그래프 구축
  - 노드: 시작 비말단 기호로부터 도출 가능한 (부분)프로그램들
  - 간선: 문법 규칙 I번 적용에 대응
    - 가중치: 문법 규칙이 적용될 확률에 -log 취한 값
- 시작 비말단 기호부터 가장 거리가 짧은 순으로 프로그램 나열
  - 이 때 최단거리 계산은 A* 알고리즘 사용



$Pr(S \to S + S) = 0.3$
$Pr(S \to x \mid S + S) = 0.8$
...
$Pr(S \to x \mid x + S) = 0.2$
$Pr(S \to y \mid x + S) = 0.6$
...

실험 벤치마크



	A	B	C	D
1	Number	Phone		
2	02082012225	020-8201-2225		
3	02072221236	020-7222-1236		
4	0208123654	020-8123-654		
5	0207236523	020-7236-523		
6	02082012222	020-8201-2222		
7				
8				

STRING: End-user Programming

205 problems

complement

```
~ 01010001110101110000000000001111
  10101110001010001111111111110000
```

bitwise and

```
  01010001110101110000000000001111
& 00110001011011100011000101101110
  00010001010001100000000000001110
```

bitwise or

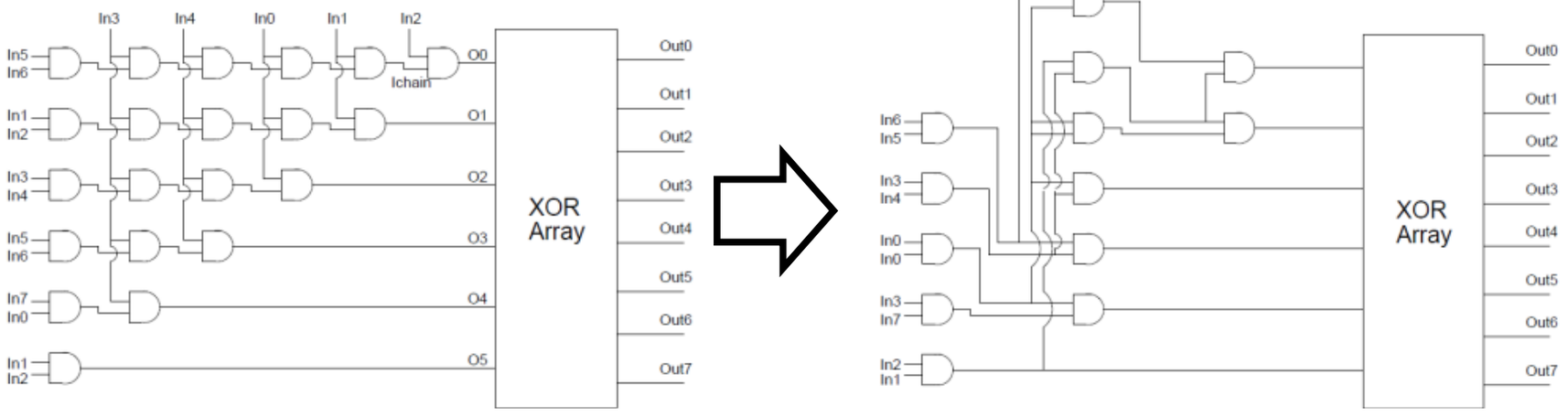
```
  01010001110101110000000000001111
| 00110001011011100011000101101110
  01110001111111110011000101101111
```

bitwise xor

```
  01010001110101110000000000001111
^ 00110001011011100011000101101110
  01100000101110010011000101100001
```

BITVEC: Efficient low-level algorithm

750 problems

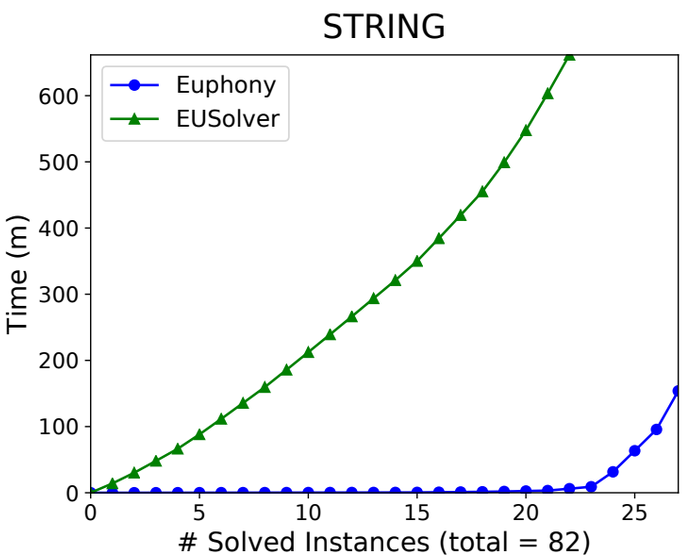


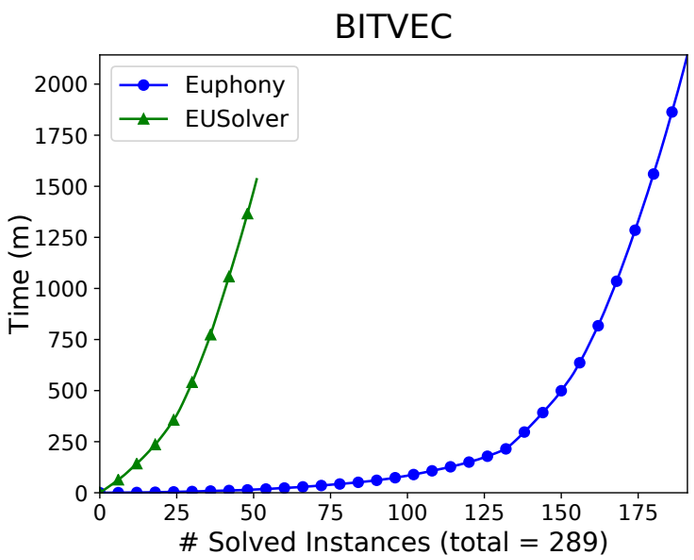
CIRCUIT: Attack-resistant crypto circuits

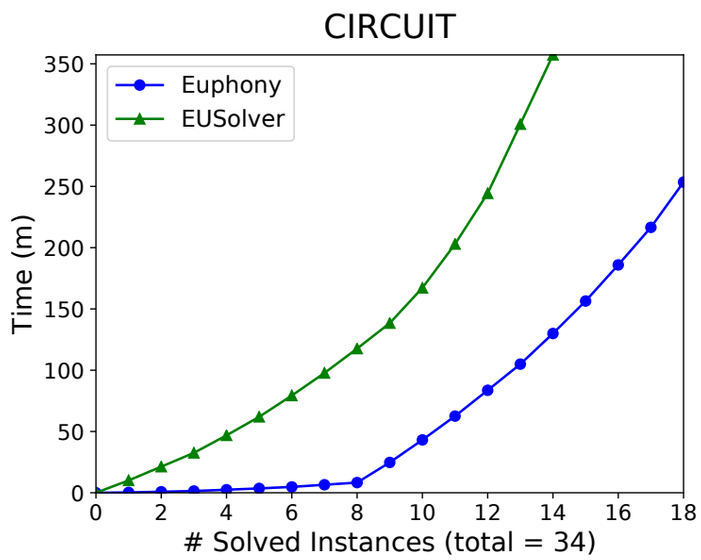
212 problems

실험 결과

- EUSolver(확률모델 가이드 없는 상향식 나열 전략)와 비교
  - 학습: EUSolver 가 10분안에 푼 문제들의 솔루션
  - 테스트셋: 405 (timeout: 1시간)







귀납 합성 전략

- 하나씩 나열해보기
  - 하향식 (Top-Down)
  - 상향식 (Bottom-Up)
    - 하향식 + 통계모델 사용 (PLDI'18)
  - **하향식 분할정복**
  - 상향식 나열 + 효율적 자료구조 쓰기 (POPL'21)

예제 합성 문제

- Goal: function f converting a given phone number x into another format
- Specification

Syntactic:

$$
\begin{aligned}
S &\rightarrow x \mid \text{“+”} \mid \text{“␣”} \mid \text{“-”} \mid \text{“.”} \mid \mathrm{ConCat}(S, S) \\
&\mid \mathrm{SubStr}(S, I, I) \\
&\mid \mathrm{IntToStr}(I) \\
&\mid \mathrm{Replace}(S, S, S) \\
I &\rightarrow 1 \mid \cdots \mid 9 \mid I - I \mid \mathrm{Length}(S)
\end{aligned}
$$

String concatenation

Semantic:

$$f(\text{“␣+1␣2.3”}) = \text{“1␣2-3.”} \wedge f(\text{“␣+4␣56.78”}) = \text{“4␣56-78.”}$$

예제 합성 문제

- Goal: function f converting a given phone number x into another format
- Syntax:

$$\begin{aligned} S &\rightarrow x \mid \sqcup \mid \text{“-”} \mid \text{“.”} \mid \text{ConCat}(S, S) \\ &\mid \text{SubStr}(S, I, I) \\ &\mid \text{IntToStr}(I) \\ &\mid \text{Replace}(S, S, S) \\ I &\rightarrow 1 \mid \cdots \mid 9 \mid I - I \mid \text{Length}(S) \end{aligned}$$

$\text{SubStr}(str, start\_pos, len)$

e.g., $\text{SubStr}(\text{“}abc\text{”}, 0, 2) = \text{“}ab\text{”}$

Semantic:

$f(\text{“}\sqcup + 1 \sqcup 2.3\text{”}) = \text{“}1 \sqcup 2\text{-}3.\text{”} \wedge f(\text{“}\sqcup + 4 \sqcup 56.78\text{”}) = \text{“}4 \sqcup 56\text{-}78.\text{”}$

예제 합성 문제

- Goal: function f converting a given phone number x into another format
- Specification

Syntactic:

$$
\begin{aligned}
S \quad &\rightarrow \quad \mathrm{Replace}(str, match, replacement) \\
&\quad\quad \text{e.g., } \mathrm{Replace}(\text{“}aba\text{”}, \text{“}a\text{”}, \text{“}b\text{”}) = \text{“}bba\text{”} \\
&\mid \quad \mathrm{IntToStr}(I) \\
&\mid \quad \mathrm{Replace}(S, S, S) \\
I \quad &\rightarrow \quad 1 \mid \cdots \mid 9 \mid I - I \mid \mathrm{Length}(S)
\end{aligned}
$$

Semantic:

$$f(\text{“}\llcorner + 1\llcorner 2.3\text{”}) = \text{“}1\llcorner 2\text{-}3.\text{”} \wedge f(\text{“}\llcorner + 4\llcorner 56.78\text{”}) = \text{“}4\llcorner 56\text{-}78.\text{”}$$

예제 합성 문제

- Solution: Size : 12 AST nodes

$$\mathrm{SubStr}(\mathrm{Replace}(\mathrm{ConCat}(x, \text{“.”}), \text{“.”}, \text{“-”}), 2, \mathrm{Length}(x) - 1)$$

하향식 분할정복 (Top-Down Propagation)

- **분할 정복**: 만약 $F(e_1, \cdots, e_k)$꼴인 프로그램이 어떤 입력에 대해서 o 를 출력한다면, $e_1, \cdots, e_k$는 같은 입력에 대해 어떤 값들을 출력해야 하는가?

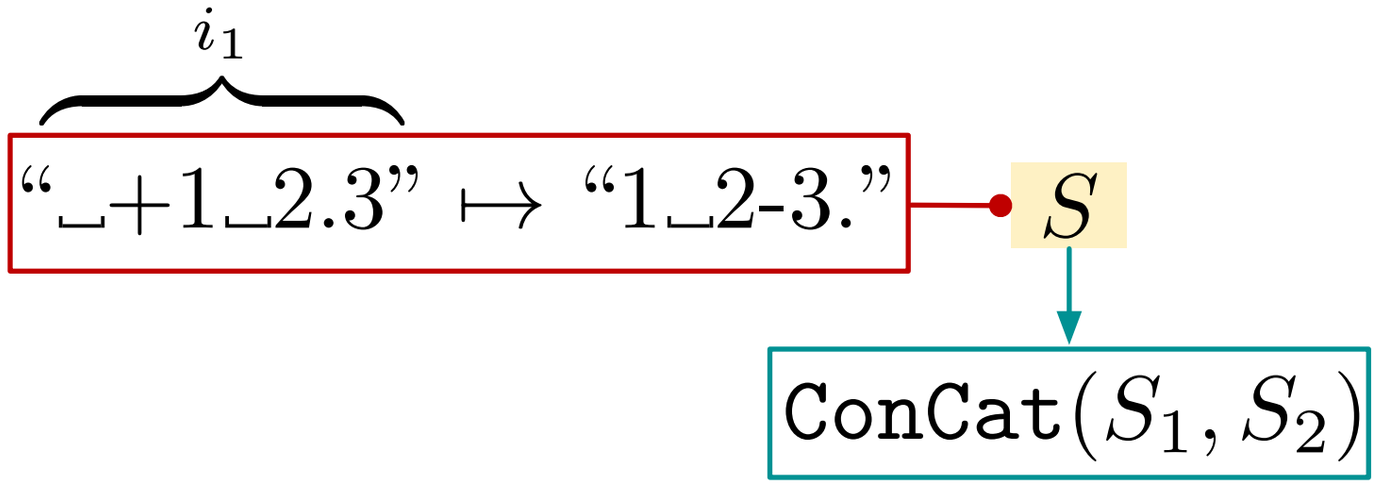
- **역함수:** 인자들이 만족시킬 입출력 조건 유추

$$F^{-1}(o) = \{(a_1, \cdots, a_k) \mid F(a_1, \cdots, a_k) = o\}$$

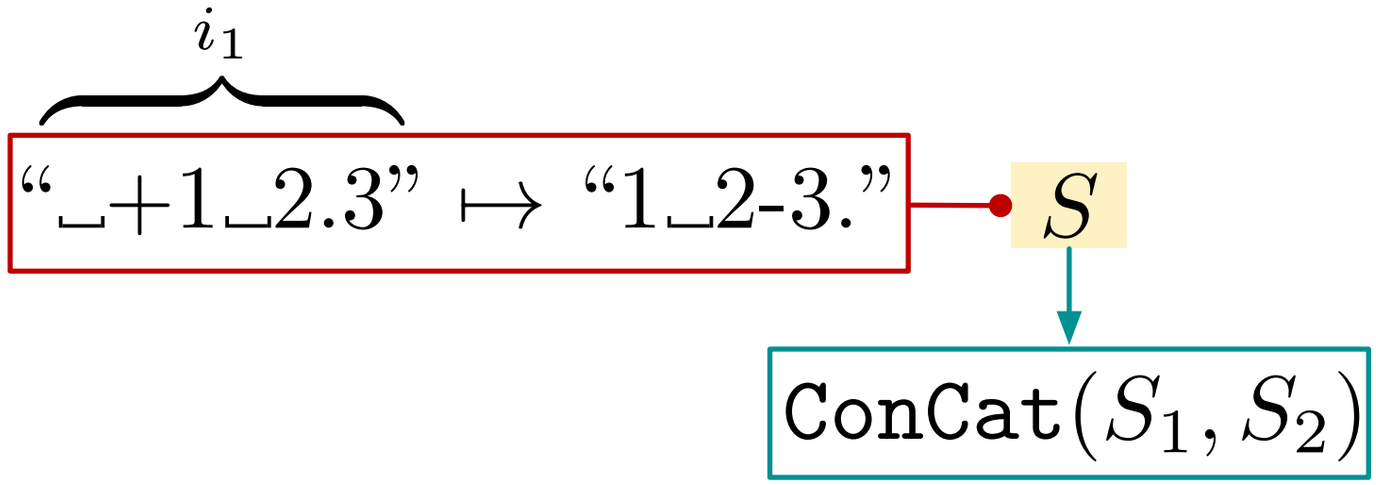
- **가능 입력들** *(inverse-set or pre-image):*

e.g., $\quad \mathrm{ConCat}^{-1}(\text{“}USA\text{”}) = \{(\text{“}U\text{”}, \text{“}SA\text{”}), (\text{“}US\text{”}, \text{“}A\text{”})\}$

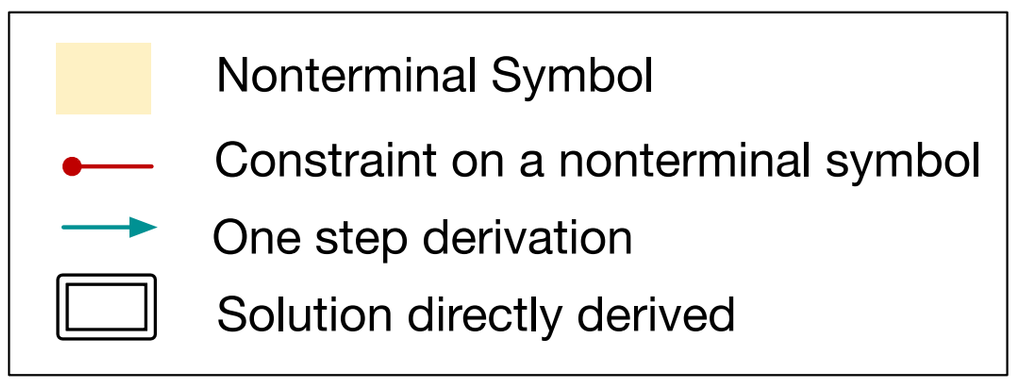
하향식 분할정복 (Top-Down Propagation)



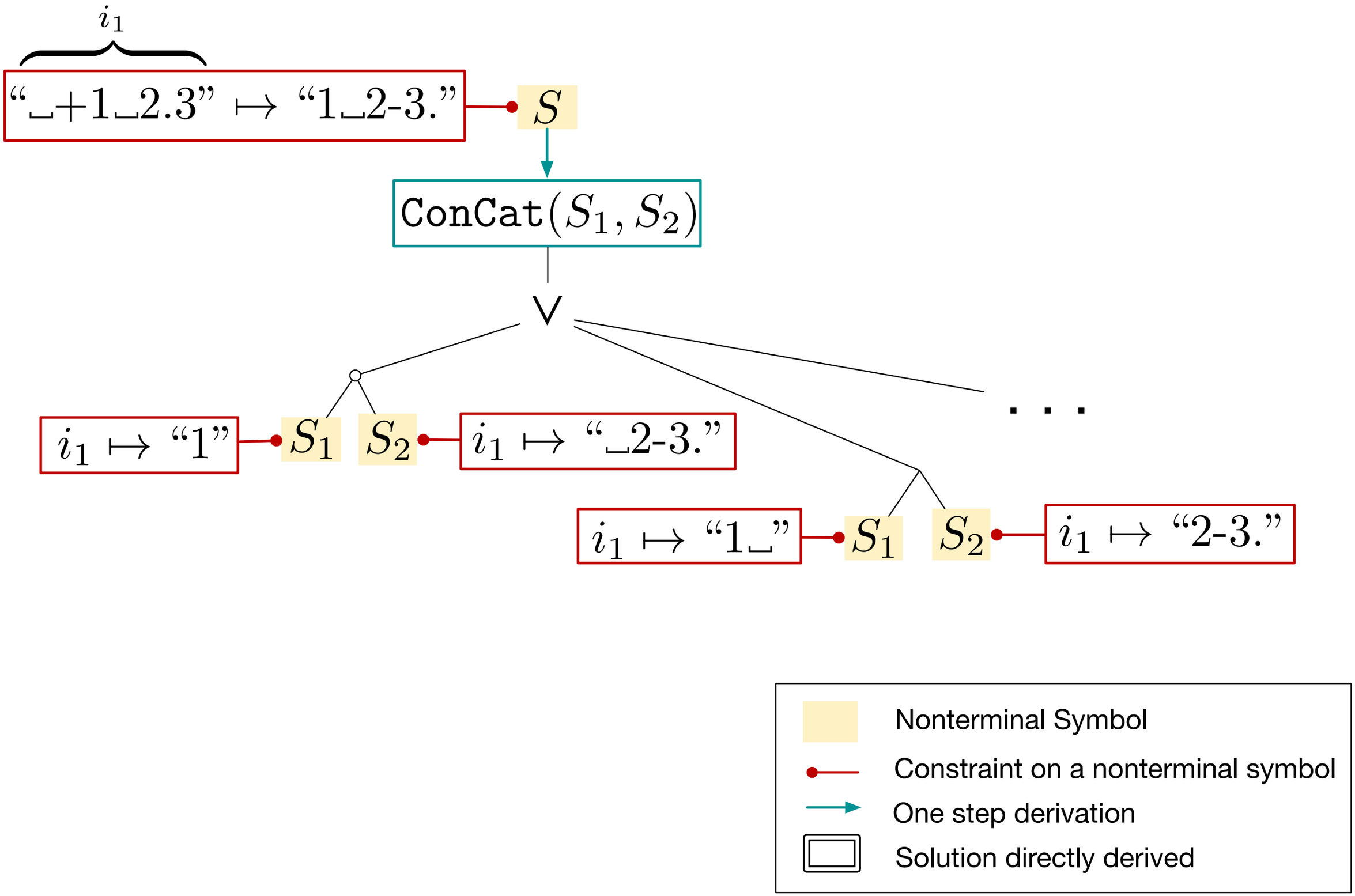
하향식 분할정복 (Top-Down Propagation)



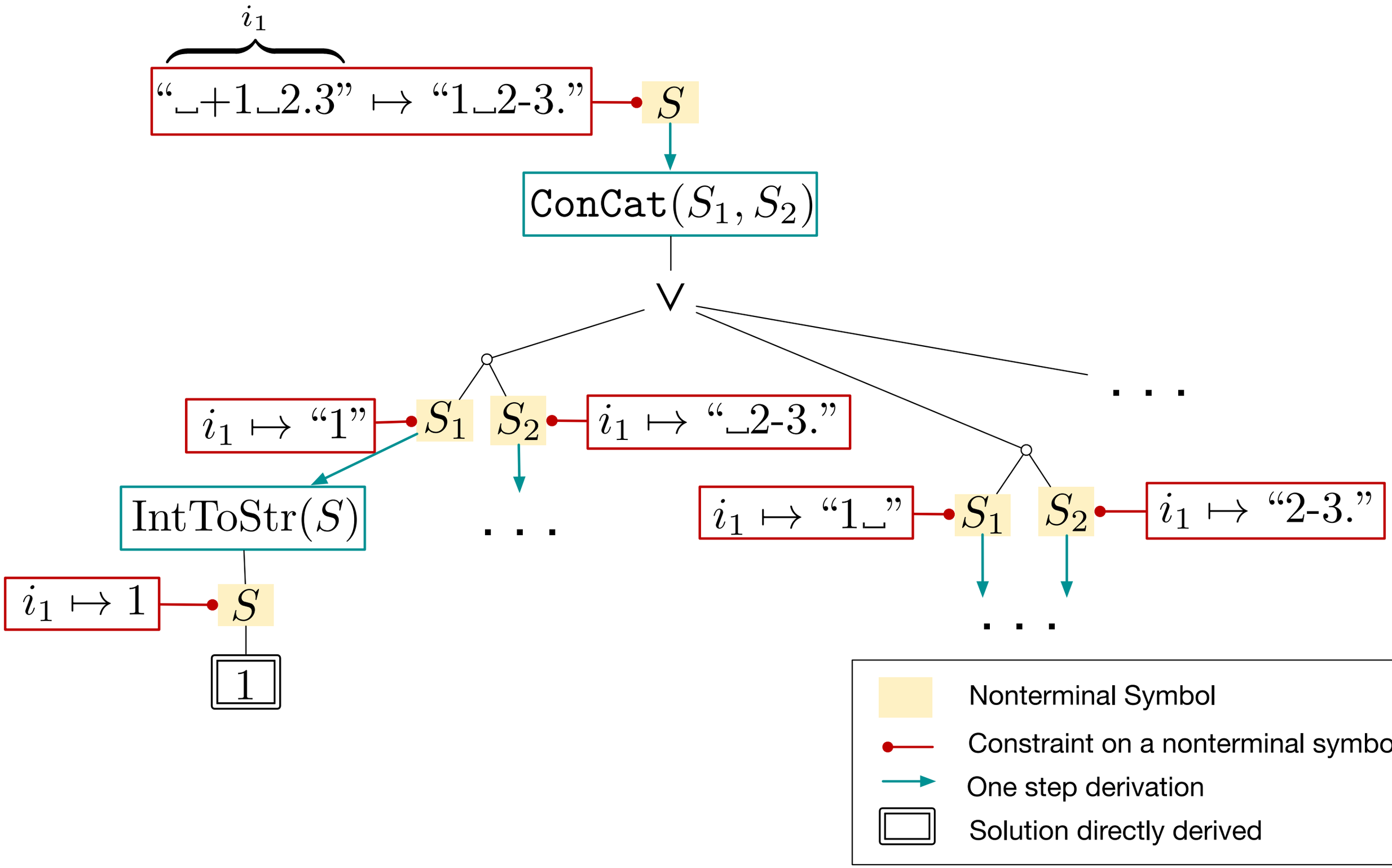
$$\text{ConCat}^{-1}(\text{“1␣2-3.”}) = \{(\text{“1”}, \text{“␣2-3.”}), (\text{“1␣”}, \text{“2-3.”}), \cdots\}$$



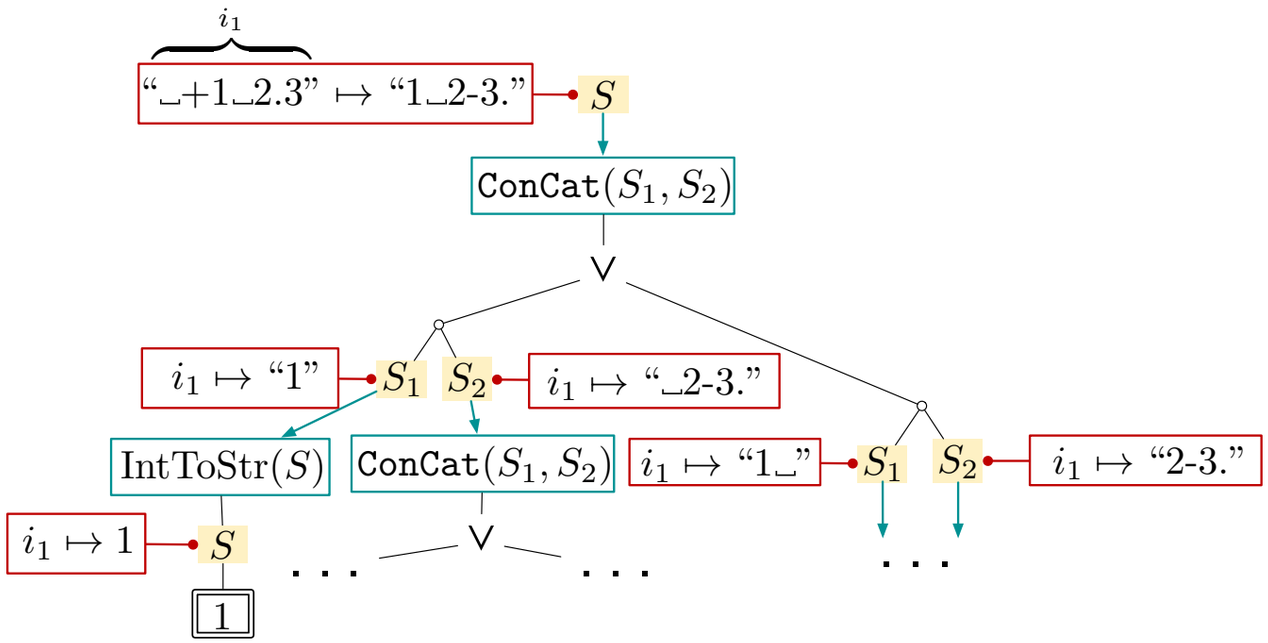
하향식 분할정복 (Top-Down Propagation)

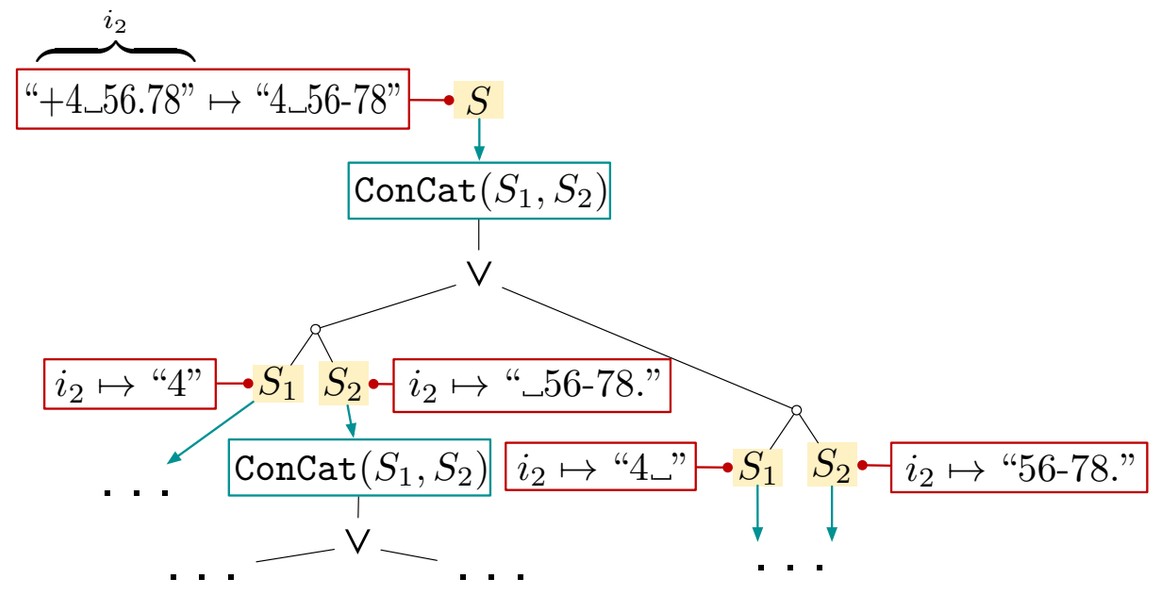


하향식 분할정복 (Top-Down Propagation)



하향식 분할정복 (Top-Down Propagation)





입출력 예제1에 맞는 모든 프로그램들

입출력 예제2에 맞는 모든 프로그램들

$\{\text{ConCat}(\text{ConCat}(\text{IntToStr}(1), \text{“␣”}), \cdots),$
$\text{ConCat}(\text{ConCat}(\text{IntToStr}(1), \text{ConCat}(\text{“␣”} \cdots),$
$\cdots\}$

$\cap$

$\{\text{ConCat}(\text{ConCat}(\text{IntToStr}(4), \text{“␣”}), \cdots),$
$\text{ConCat}(\text{ConCat}(\text{IntToStr}(4), \text{ConCat}(\text{“␣”} \cdots),$
$\cdots\}$

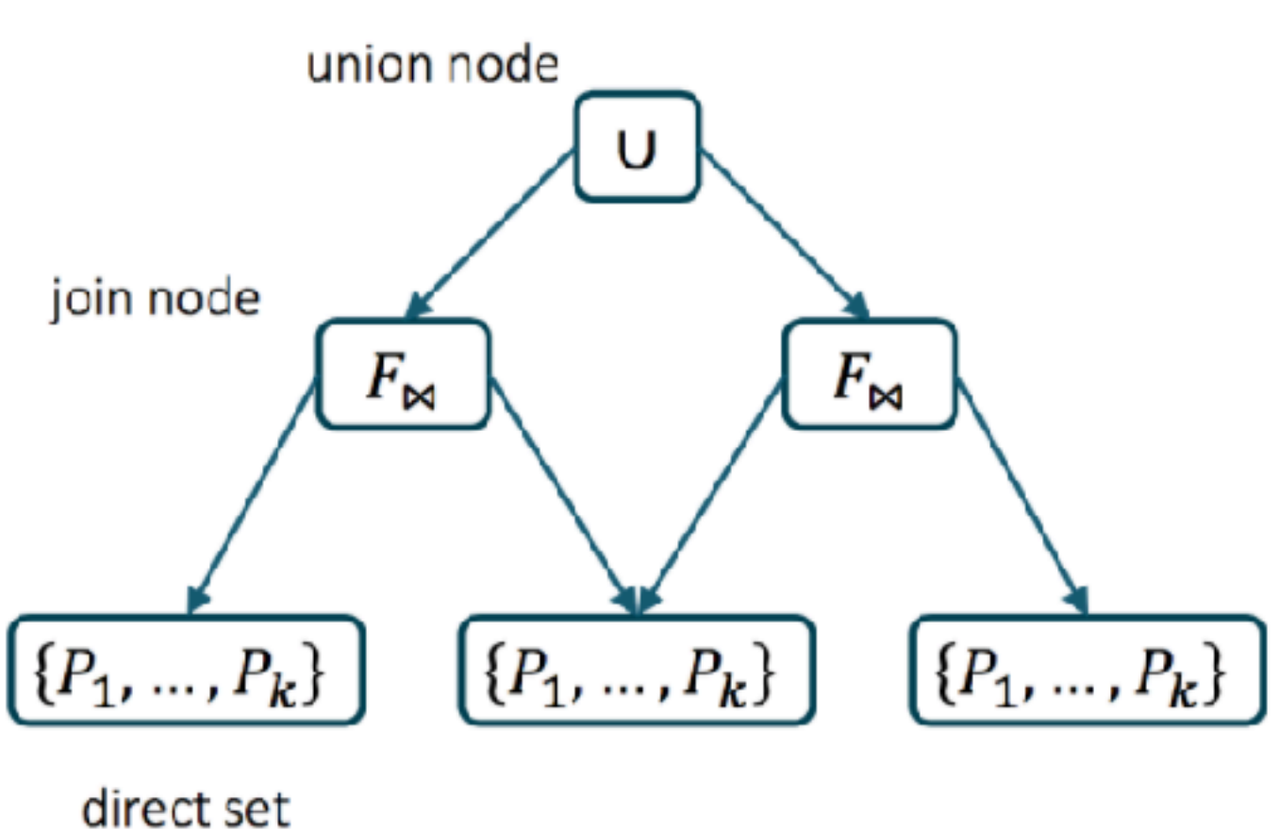
공간 효율적 자료구조에 저장됨.

최종 솔루션 집합

하향식 분할정복 + 효율적 자료구조

효율적 자료구조: 버전 공간 표현 (Version Space Algebra)

기계학습에서: 입출력 예제
에 맞는 모델 집합



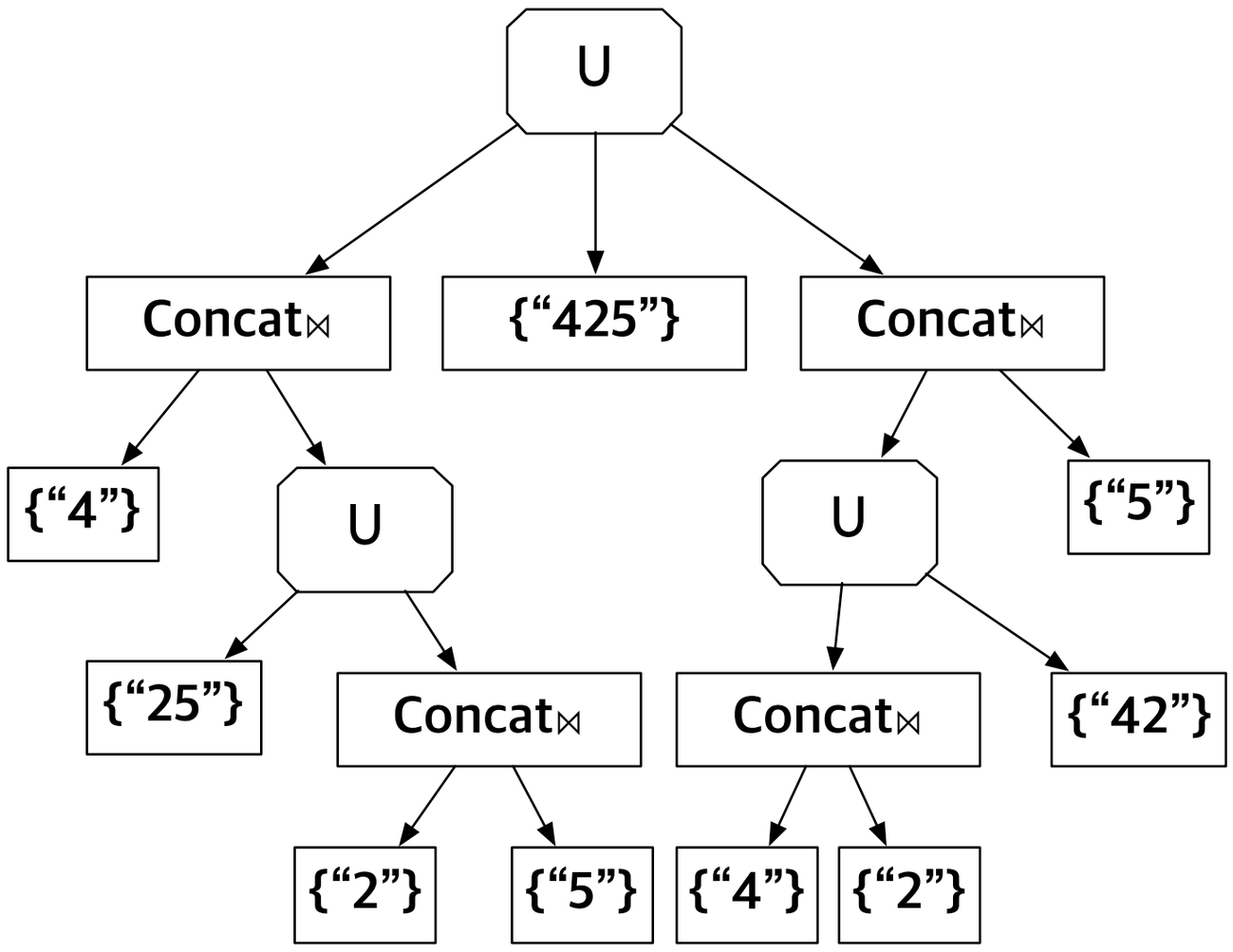
Volume of a VSA
(the number of nodes) $V(VSA)$

Size of a VSA
(the number of programs) $|VSA|$

$$V(VSA) = O(\log|VSA|)$$

버전 공간 표현 (Version Space Algebra)

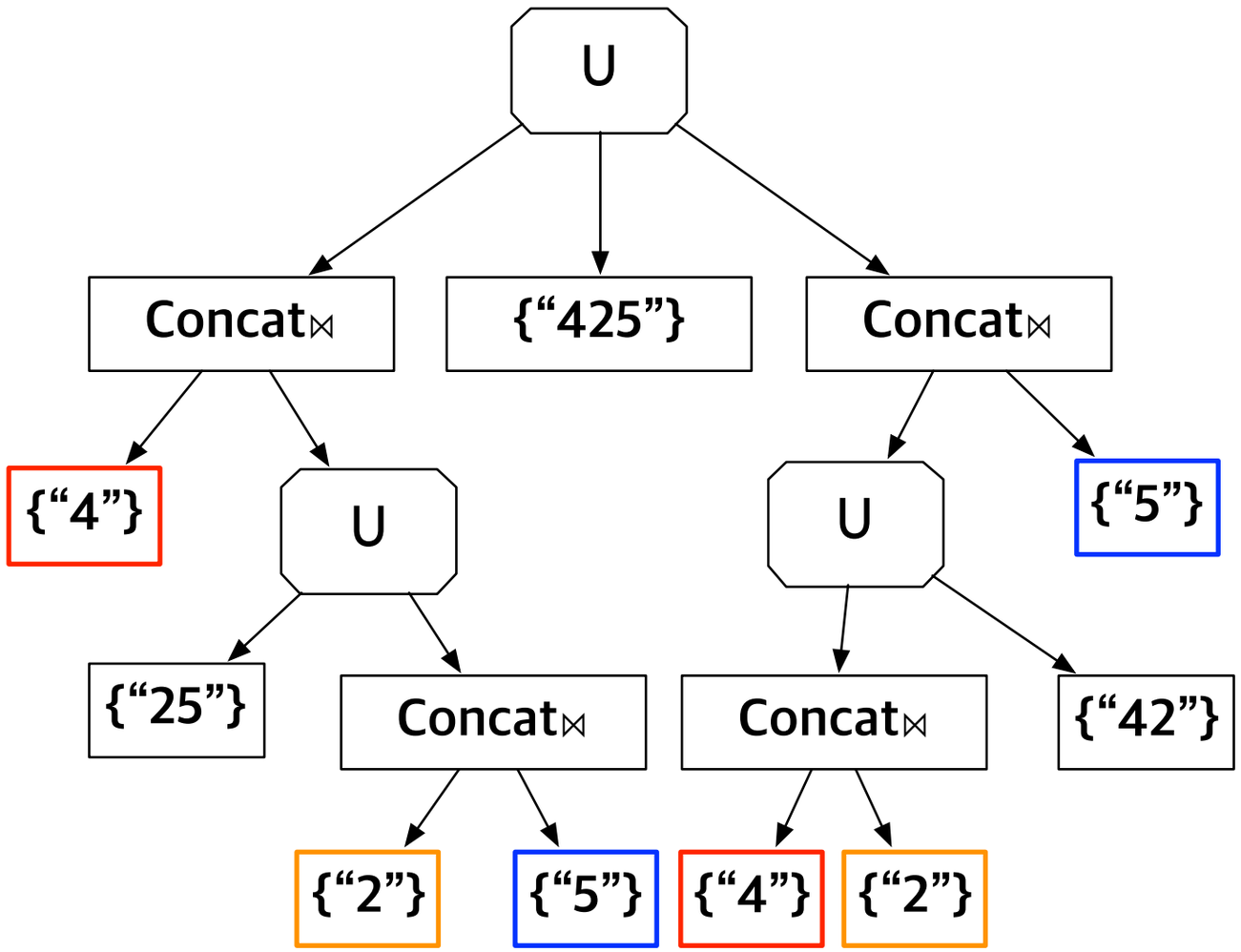
“425”를 출력하는 프로그램 집합을 표현하는 VSA



$\mathbf{Concat}_{\bowtie}(\mathbf{VS}_1, \mathbf{VS}_2) = \{\mathbf{Concat}(P_1, P_2) \mid P_1 \in \mathbf{VS}_1 , P_2 \in \mathbf{VS}_2\}$

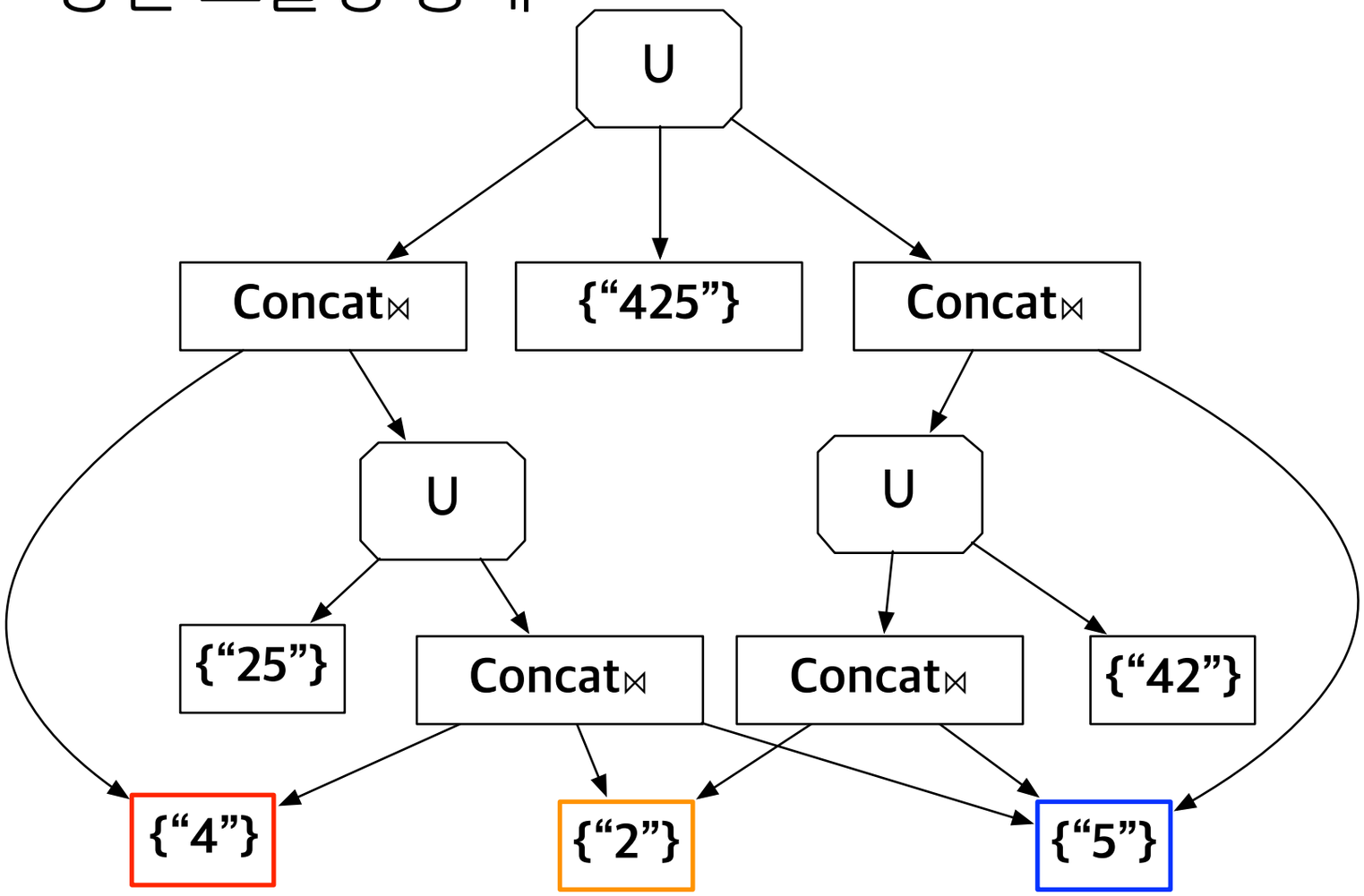
버전 공간 표현 (Version Space Algebra)

중복된 노드들 다수 존재

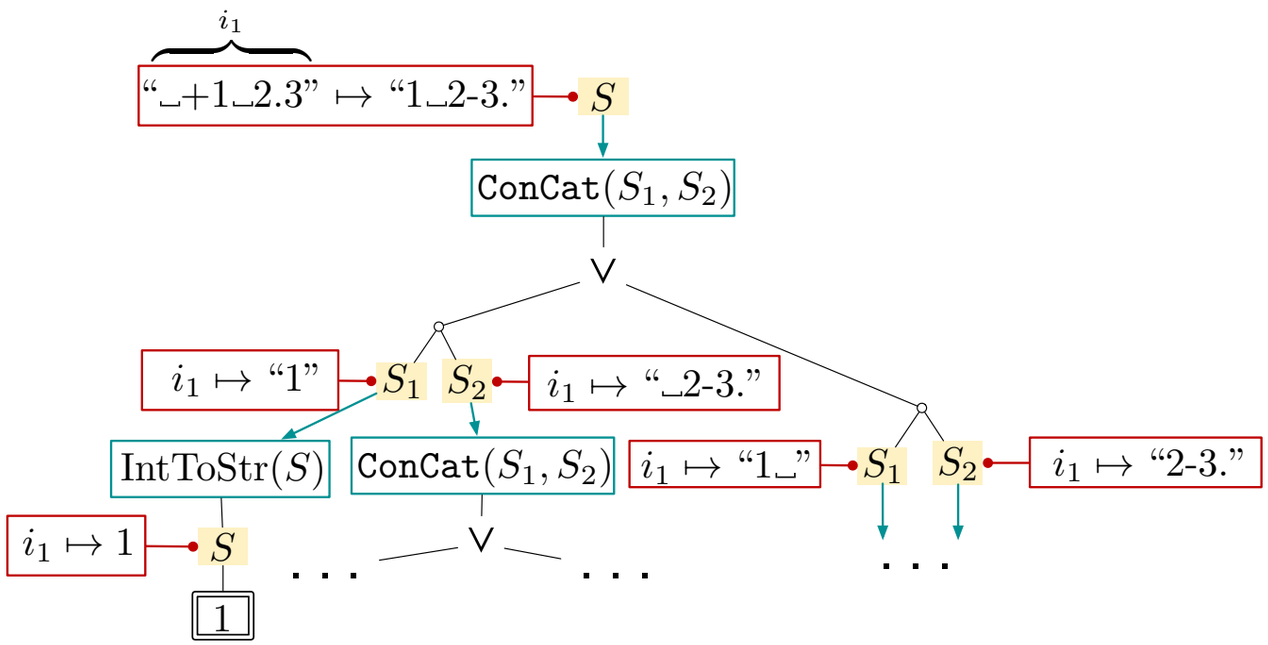


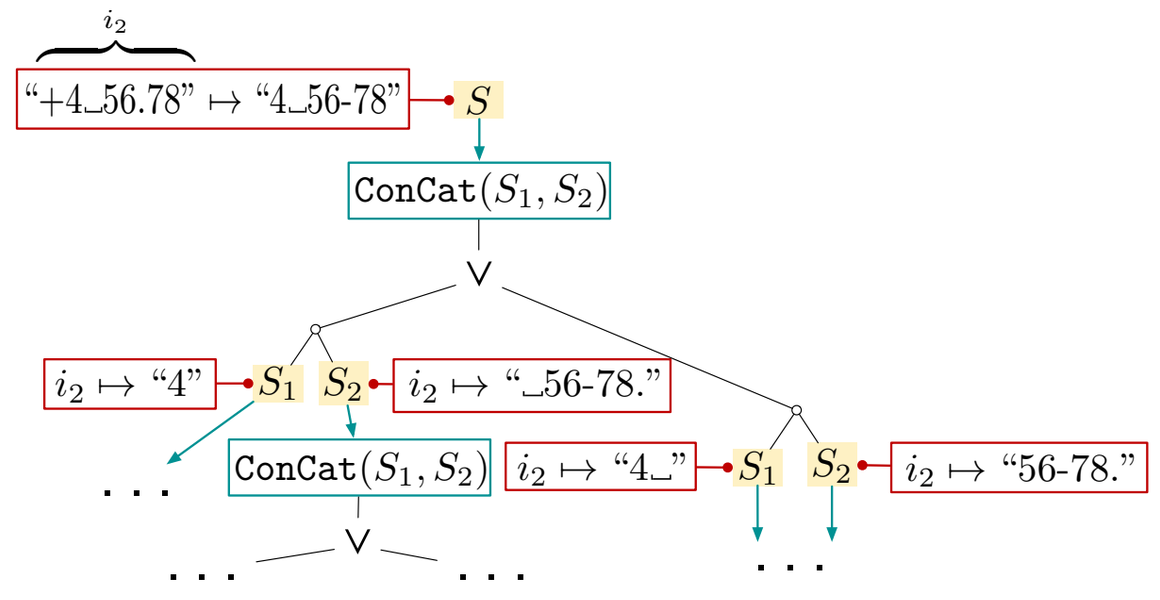
버전 공간 표현 (Version Space Algebra)

중복 제거 → 공간 효율성 증대



하향식 분할정복 (Top-Down Propagation)





입출력 예제1에 맞는 모든 프로그램들

입출력 예제2에 맞는 모든 프로그램들

{( ),

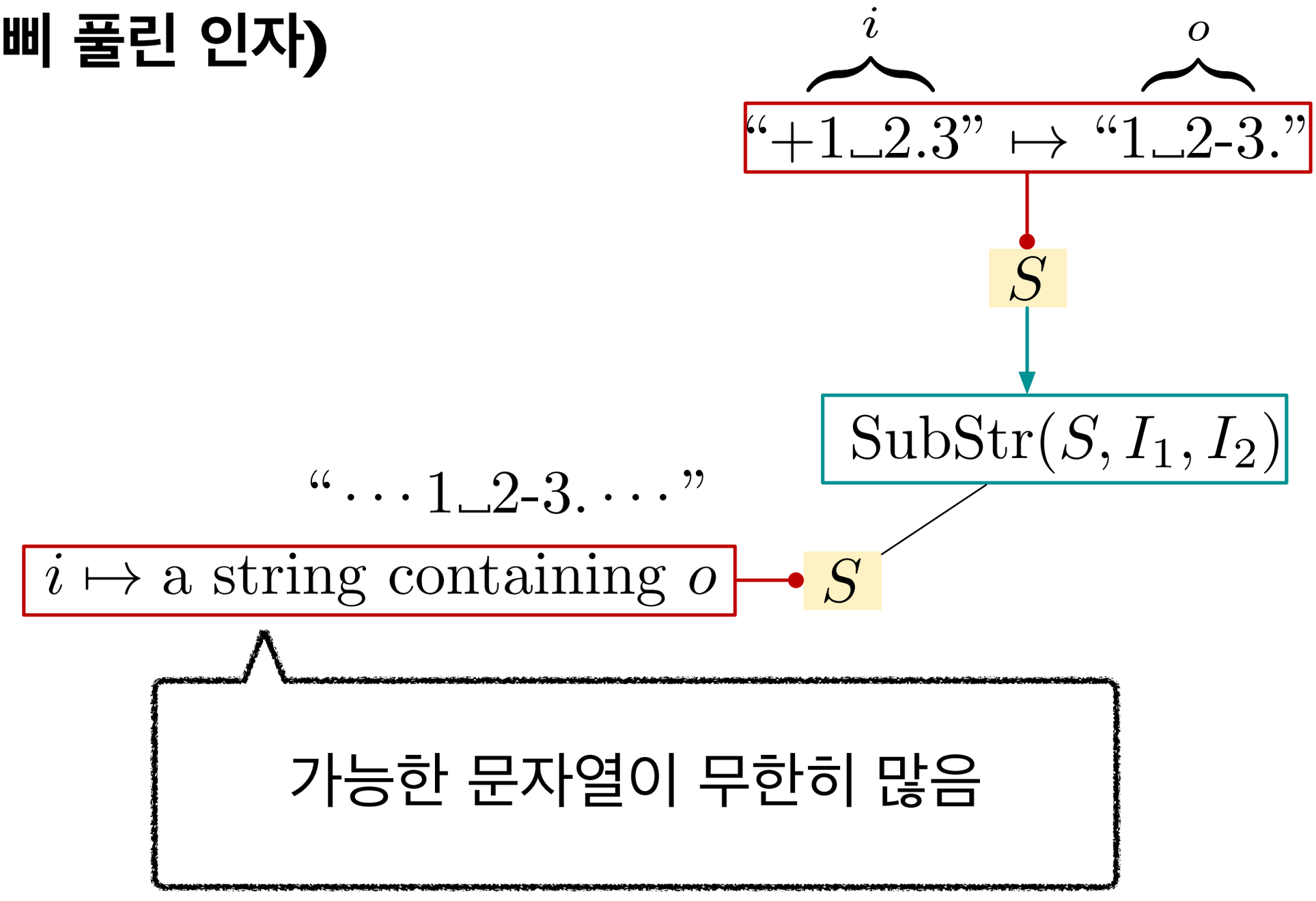
+ 매우 효율적 (분할정복, 최종목표 달성에 필요한 문제들만 풂)

– 입출력 예제 갯수↑ → 성능 ↓

– **항상 적용 가능하지는 않음**

하향식 분할정복의 문제점

가능한 입력이 무한한, 자유도가 지나친 인자들
(고삐 풀린 인자)

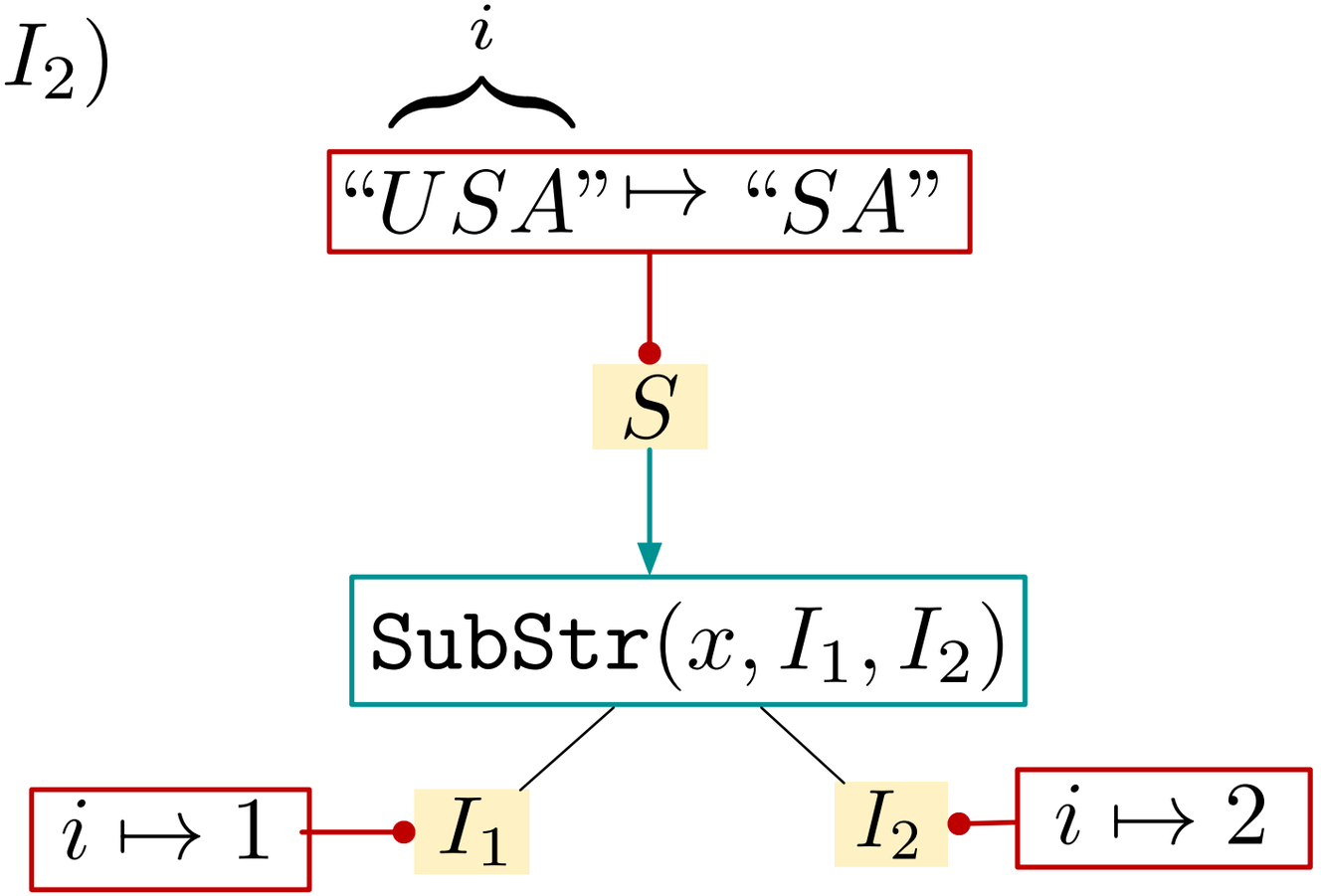


하향식 분할정복의 문제점

기존의 미봉책: 문법 제한하기

$$S \to \cdots \mid \mathrm{SubStr}(x, I_1, I_2)$$

Input example



DSL 정의에 “장인정신”이 필요 :(

“DSL design = Art + *Lots* of iterations”

“The DSL should be *expressive enough* to represent various tasks… and *restricted enough* to allow efficient search.”

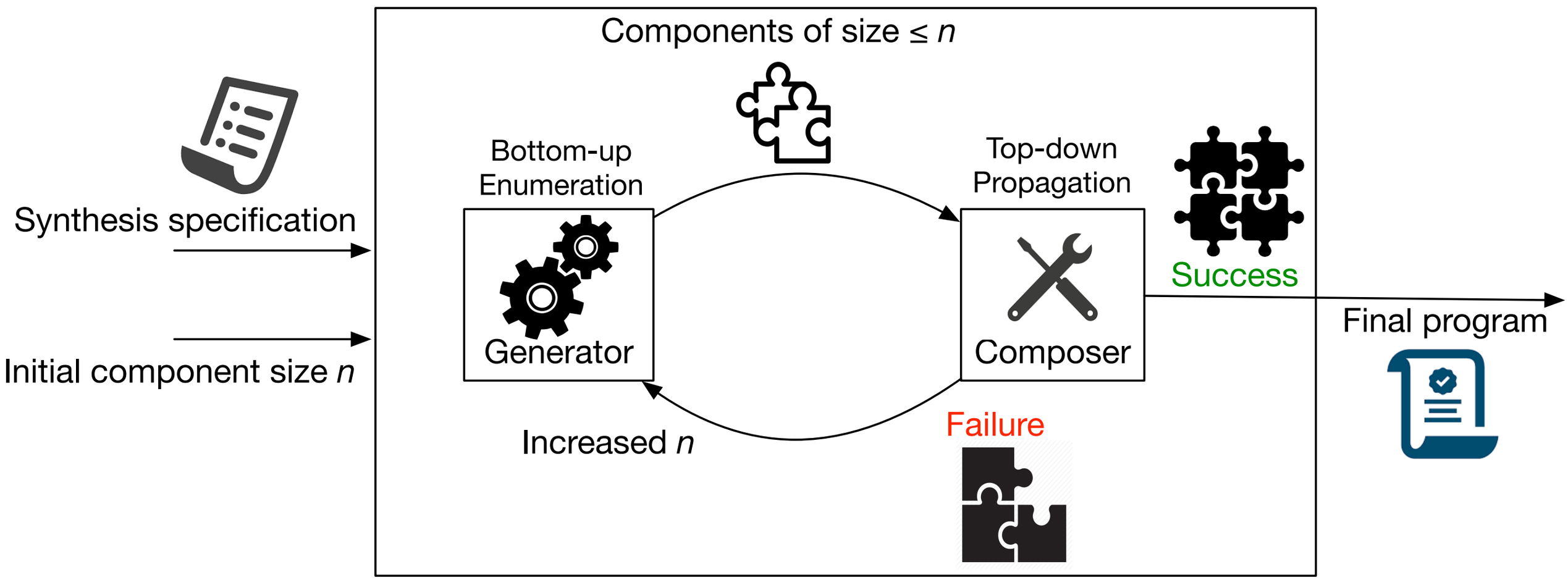
Gulwani et al., Program synthesis
Polozov et al., FlashMeta: a framework for inductive program synthesis
Gulwani et al., Programming by example (and its application to data wrangling)

귀납 합성 전략

- 하나씩 나열해보기
  - 하향식 (Top-Down)
  - 상향식 (Bottom-Up)
    - 하향식 + 통계모델 사용 (PLDI'18)
- 하향식 분할정복
- **상향식 나열 + 효율적 자료구조 쓰기 (POPL'21)**

DUET: 양방향식 프로그램 합성†

하향식 분할정복(일반성 부족), 상향식 나열(느린 성능)의 한계 모두 해결.



† Domain-unaware inductive synthesis combining Enumeration and Top-down Propagation

아이디어

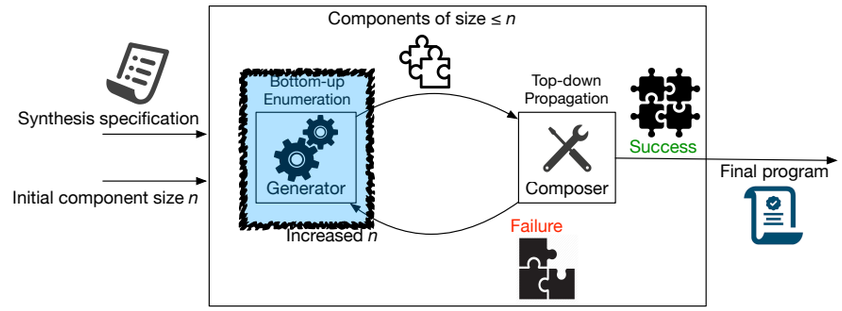
- 상향식 나열은 느리지만 "핵심"부분 표현식들은 빨리 찾을 수 있음.

$$\mathrm{SubStr}(\boxed{\mathrm{Replace}(\mathrm{ConCat}(x, \text{“.”}), \text{“.”}, \text{“-”})}, 2, \mathrm{Length}(x) - 1)$$

고삐풀린 인자들 자리에 들어갈 표현식

- 우리의 하향식 분할정복: 고삐 풀린 인자들 자리에 상향식 나열로 만든 프로그램들 끼워넣기.

부품 생성

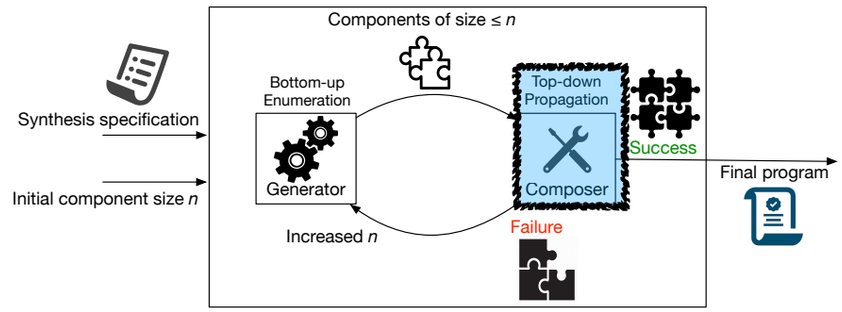


처음 부품 크기가 6으로 주어졌다고 가정

상향식 나열로 다음 부품들 생성

$$C = \left\{ \begin{array}{c} x, \text{“+”}, \text{“.”}, \text{“␣”}, \text{“-”}, \\ 1, \cdots, \text{Length}(x), \\ \text{ConCat}(x, \text{“.”}), \\ \cdots \\ \text{Replace}(\text{ConCat}(x, \text{“.”}), \text{“.”}, \text{“}-\text{”}) \end{array} \right\}$$

가능입력들 ⊆ 부품의 출력들



Component Pool

$$C = \left\{ \begin{array}{c} x, \text{“+”}, \text{“.”}, \text{“␣”}, \text{“-”}, \\ 1, \cdots, \text{Length}(x), \\ \text{ConCat}(x, \text{“.”}), \\ \cdots \\ \text{Replace}(\text{ConCat}(x, \text{“.”}), \text{“.”}, \text{“}-\text{”}) \end{array} \right\}$$

$$\overbrace{\text{“␣+1␣2.3”}}^{i_1} \mapsto \overbrace{\text{“1␣2-3.”}}^{o_1}$$
$$\underbrace{\text{“␣+4␣56.78”}}_{i_2} \mapsto \underbrace{\text{“4␣56-78.”}}_{o_2}$$

S

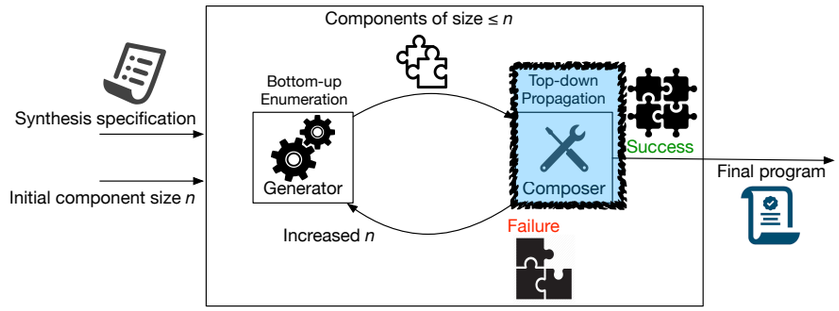
$\text{SubStr}(S, I_1, I_2)$

S

$i_1 \mapsto [\![e]\!](i_1) \quad i_2 \mapsto [\![e]\!](i_2)$
where $e \in C$, $o_1 \prec [\![e]\!](i_1)$,
$o_2 \prec [\![e]\!](i_2)$

출력 문자열들을 ‘포함’하는 문자열들을 생성하는 부품 C에서 찾아서 있으면 쓰기

가능입력들 ⊆ 부품의 출력들



Component Pool

$$C = \left\{ \begin{array}{c} x, \text{“+”}, \text{“.”}, \text{“␣”}, \text{“-”}, \\ 1, \cdots, 9 \end{array} \right\}$$

$$S$$

$$\begin{array}{cc} \overbrace{\text{“␣+1␣2.3”}}^{i_1} & \overbrace{\mapsto \text{“1␣2-3.”}}^{o_1} \\ \underbrace{\text{“␣+4␣56.78”}}_{i_2} & \underbrace{\mapsto \text{“4␣56-78.”}}_{o_2} \end{array}$$

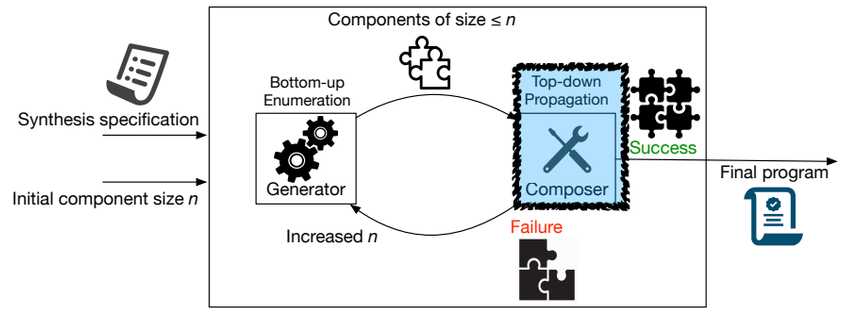
$$\text{SubStr}(S, I_1, I_2)$$

$$\begin{array}{ll} i_1 \mapsto [\![e]\!](i_1) & i_2 \mapsto [\![e]\!](i_2) \\ \text{where } e \in C, & o_1 \prec [\![e]\!](i_1), \\ & o_2 \prec [\![e]\!](i_2) \end{array}$$

부분 문제 생성시 모든 입출력 예제 한꺼번에 고려:

- 최종 교집합 연산 필요 없음
- 입출력 예제 갯수가 늘어나도 성능 부하 적음.

가능입력들 ⊆ 부품의 출력들



Component Pool

$$C = \left\{ \begin{array}{c} x, \text{“+”}, \text{“.”}, \text{“␣”}, \text{“-”}, \\ 1, \cdots, \text{Length}(x), \\ \text{ConCat}(x, \text{“.”}), \\ \cdots \\ \text{Replace}(\text{ConCat}(x, \text{“.”}), \text{“.”}, \text{“}-\text{”}) \end{array} \right\}$$

$$\overbrace{\text{“␣+1␣2.3”}}^{i_1} \mapsto \overbrace{\text{“1␣2-3.”}}^{o_1}$$
$$\underbrace{\text{“␣+4␣56.78”}}_{i_2} \mapsto \underbrace{\text{“4␣56-78.”}}_{o_2}$$

S

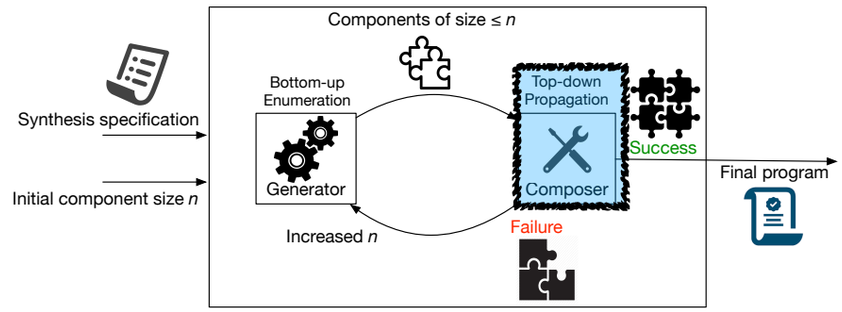
$\text{SubStr}(S, I_1, I_2)$

S

$i_1 \mapsto [\![e]\!](i_1) \quad i_2 \mapsto [\![e]\!](i_2)$
where $e \in C$, $o_1 \prec [\![e]\!](i_1)$,
$o_2 \prec [\![e]\!](i_2)$

$\text{Replace}(\text{ConCat}(x, \text{“.”}), \text{“.”}, \text{“}-\text{”})$
가 바로 그런 부품! (입력 1,2에 대해 각각
“ +1 2-3.”, “ +4 56-78.” 출력)

가능입력들 ⊆ 부품의 출력들



Component Pool

$$C = \left\{ \begin{array}{c} x, \text{“+”}, \text{“.”}, \text{“␣”}, \text{“-”}, \\ 1, \cdots, \text{Length}(x), \\ \text{ConCat}(x, \text{“.”}), \\ \cdots \\ \text{Replace}(\text{ConCat}(x, \text{“.”}), \text{“.”}, \text{“}-\text{”}) \end{array} \right\}$$

$$\begin{array}{l} i_1 \mapsto [\![e]\!](i_1) \quad i_2 \mapsto [\![e]\!](i_2) \\ \text{where } e \in C, \ o_1 \prec [\![e]\!](i_1), \\ \qquad\qquad\ o_2 \prec [\![e]\!](i_2) \end{array}$$

S

$\text{SubStr}(S, I_1, I_2)$

S

i_1 o_1

“␣+1␣2.3” ↦ “1␣2-3.”
“␣+4␣56.78” ↦ “4␣56-78.”

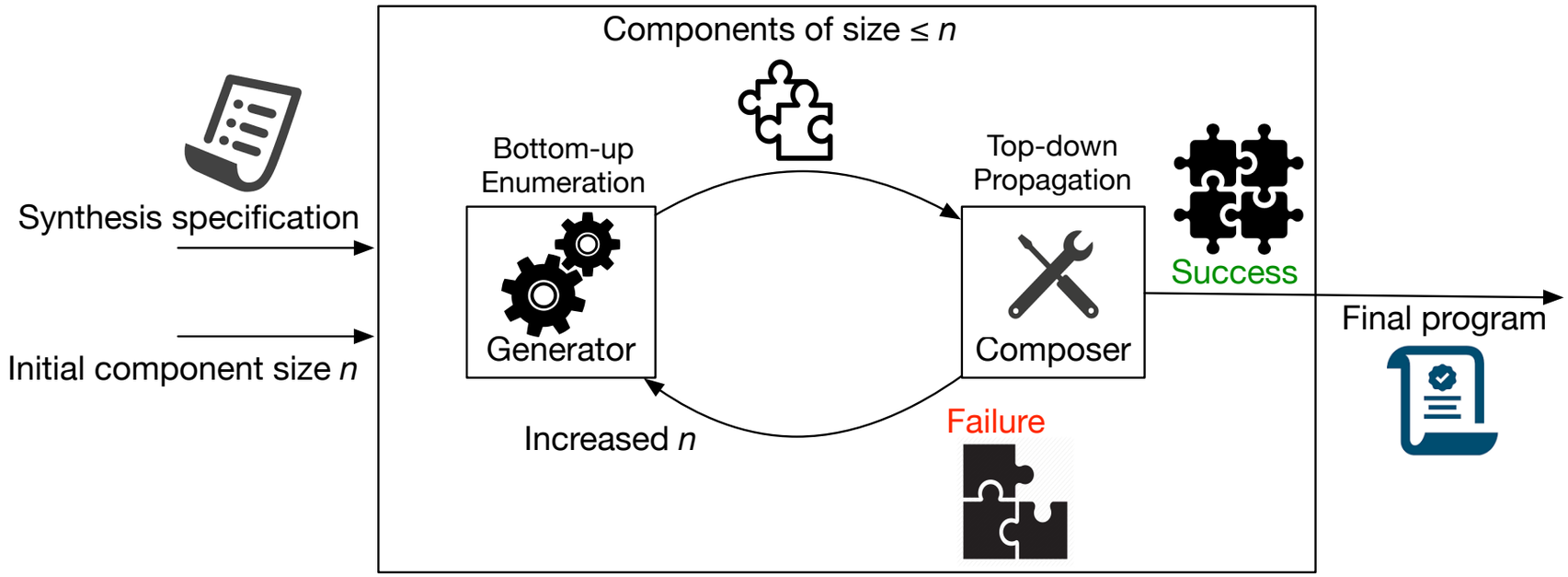
i_2 o_2

첫번째 인자가 정해진 후, 나머지 인자에 적절한 부품들 또한 C에서 찾음.

(I_1 은 2 , I_2 은 $\text{Length}(x) - 1$)

탐색 완전성

- 자유로운 인자 자리에 부품값만 들어갈 수 있음에도 불구
- 해가 탐색 공간에 존재할 경우, 이를 항상 찾아냄



- 부품 크기를 계속 늘리다 보면 솔루션이 포함될 것이기 때문

Evaluation Setup

- Benchmarks: **1,536** SyGuS problems
  - **1,167** from the SyGuS annual competitions + **369** from optimization tasks for homomorphic evaluation [Lee et al. PLDI'20]
- Comparison to three baselines (Timeout 1 hour):
  - EUSolver: winner of 2016 SyGuS competition
  - CVC4: winner of 2017 - 2019 SyGuS competition
  - Euphony [Lee et al. PLDI'18]: statistical model-guided synthesizer

실험 벤치마크

- **1,536** SyGuS 문제들
  - 경진대회 **1,167** 문제 + 프로그램 합성 기반 동형암호 프로그램 최적화기 에서 **369** 문제 [Lee et al. PLDI'20]
- 경쟁 대상들
  - EUSolver: 2016 SyGuS 경진대회 우승
  - CVC4: 2017 년 이래 계속 SyGuS 경진대회 우승
  - Euphony [Lee et al. PLDI'18]: 통계모델 기반 합성기

실험 벤치마크

	A	B
1	Email	Column 2
2	Nancy.FreeHafer@fourthcoffee.com	nancy freehafer
3	Andrew.Cencici@northwindtraders.com	andrew cencici
4	Jan.Kotas@litwareinc.com	jan kotas
5	Mariya.Sergienko@gradicdesigninstitute.com	mariya sergienko
6	Steven.Thorpe@northwindtraders.com	steven thorpe
7	Michael.Neipper@northwindtraders.com	michael neipper
8	Robert.Zare@northwindtraders.com	robert zare
9	Laura.Giussani@adventure-works.com	laura giussani
10	Anne.HL@northwindtraders.com	anne hl
11	Alexander.David@contoso.com	alexander david
12	Kim.Shane@northwindtraders.com	kim shane
13	Manish.Chopra@northwindtraders.com	manish chopra
14	Gerwald.Oberleitner@northwindtraders.com	gerwald oberleitner
15	Amr.Zaki@northwindtraders.com	amr zaki
16	Yvonne.McKay@northwindtraders.com	yvonne mckay
17	Amanda.Pinto@northwindtraders.com	amanda pinto

STRING: End-user programming

205 problems

complement

```
~ 01010001110101110000000000001111
  10101110001010001111111111110000
```

bitwise and

```
  01010001110101110000000000001111
& 00110001011011100011000101101110
  00010001010001100000000000001110
```

bitwise or

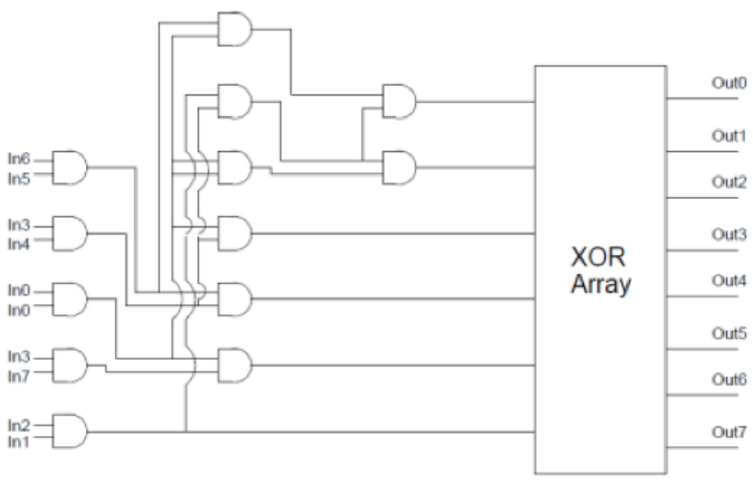
```
  01010001110101110000000000001111
| 00110001011011100011000101101110
  01110001111111110011000101101111
```

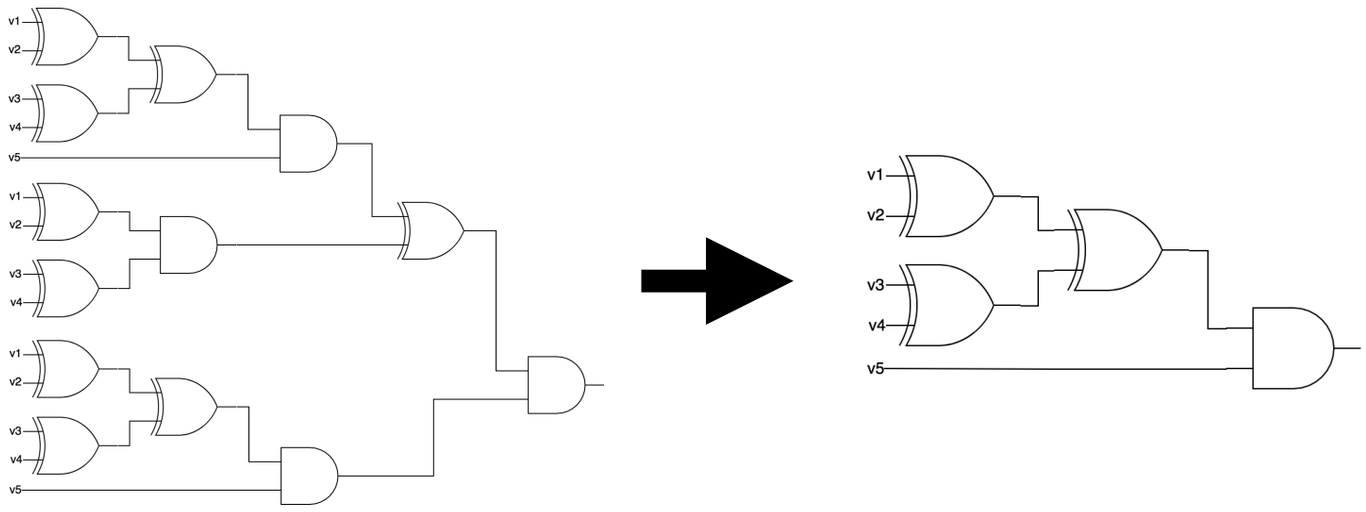
bitwise xor

```
  01010001110101110000000000001111
^ 00110001011011100011000101101110
  01100000101110010011000101100001
```

BITVEC: Efficient low-level algorithm

750 problems

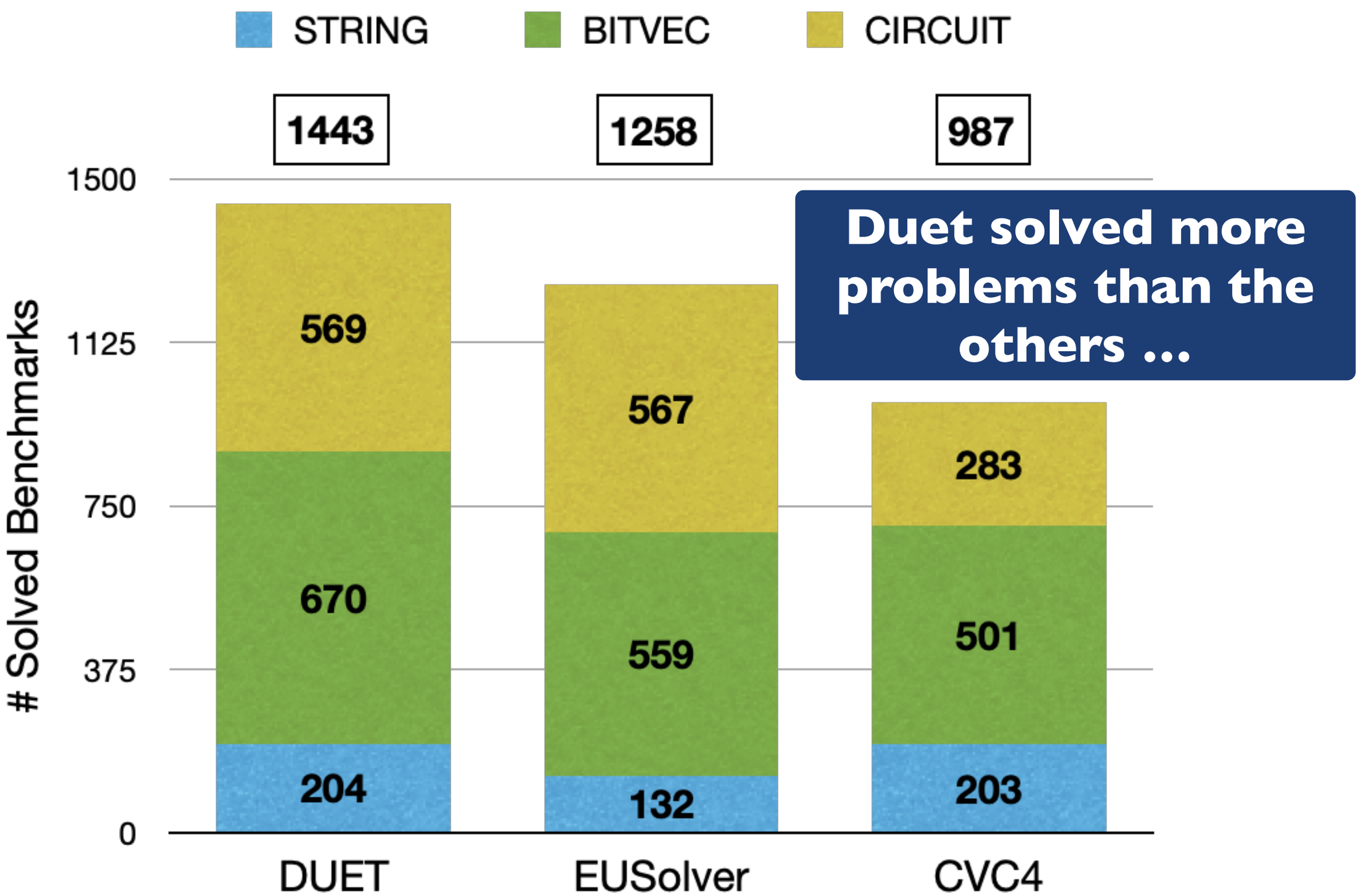




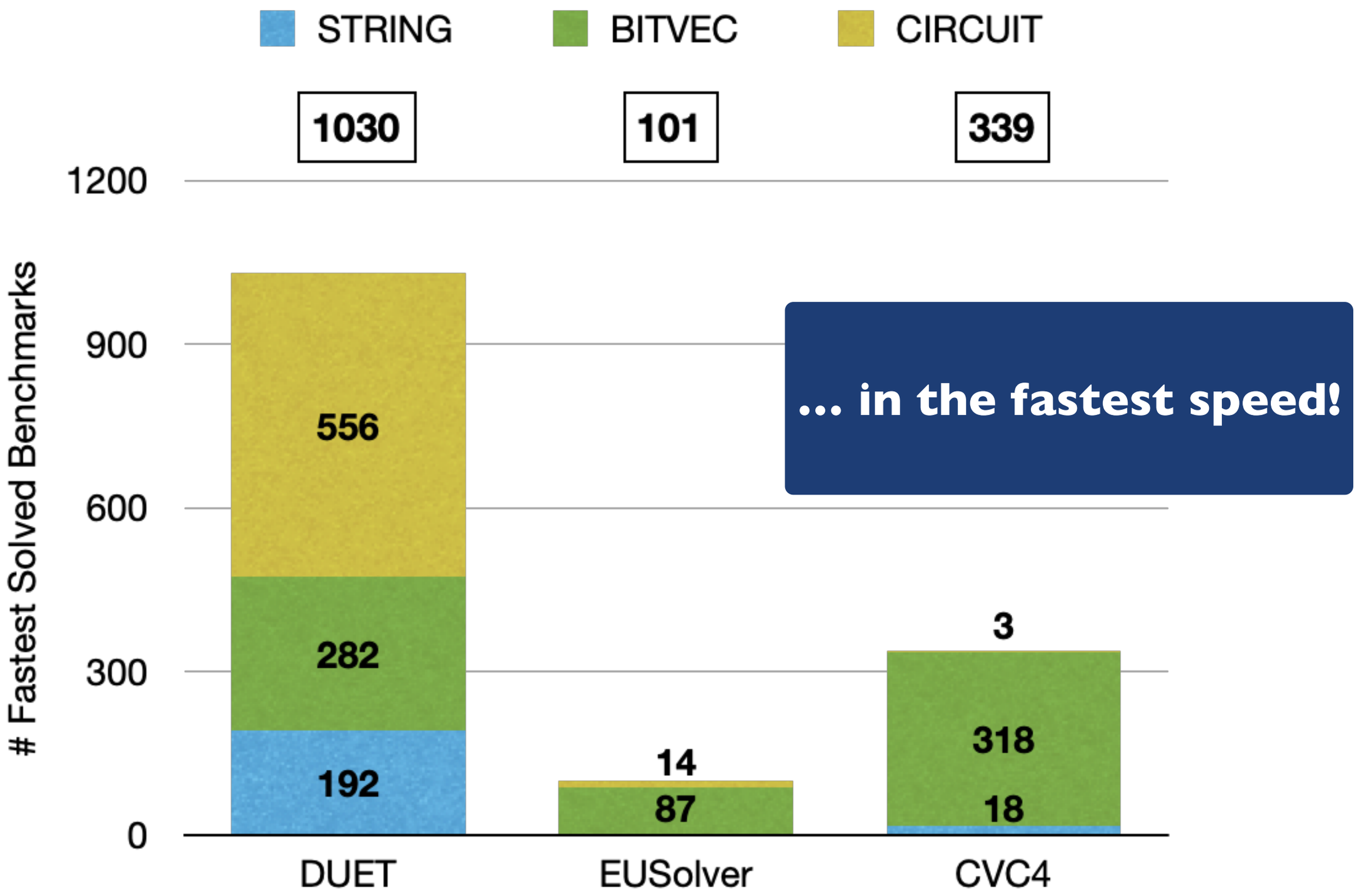
CIRCUIT: Attack-resilient crypto circuits + Optimized homomorphic evaluation circuits

581 problems

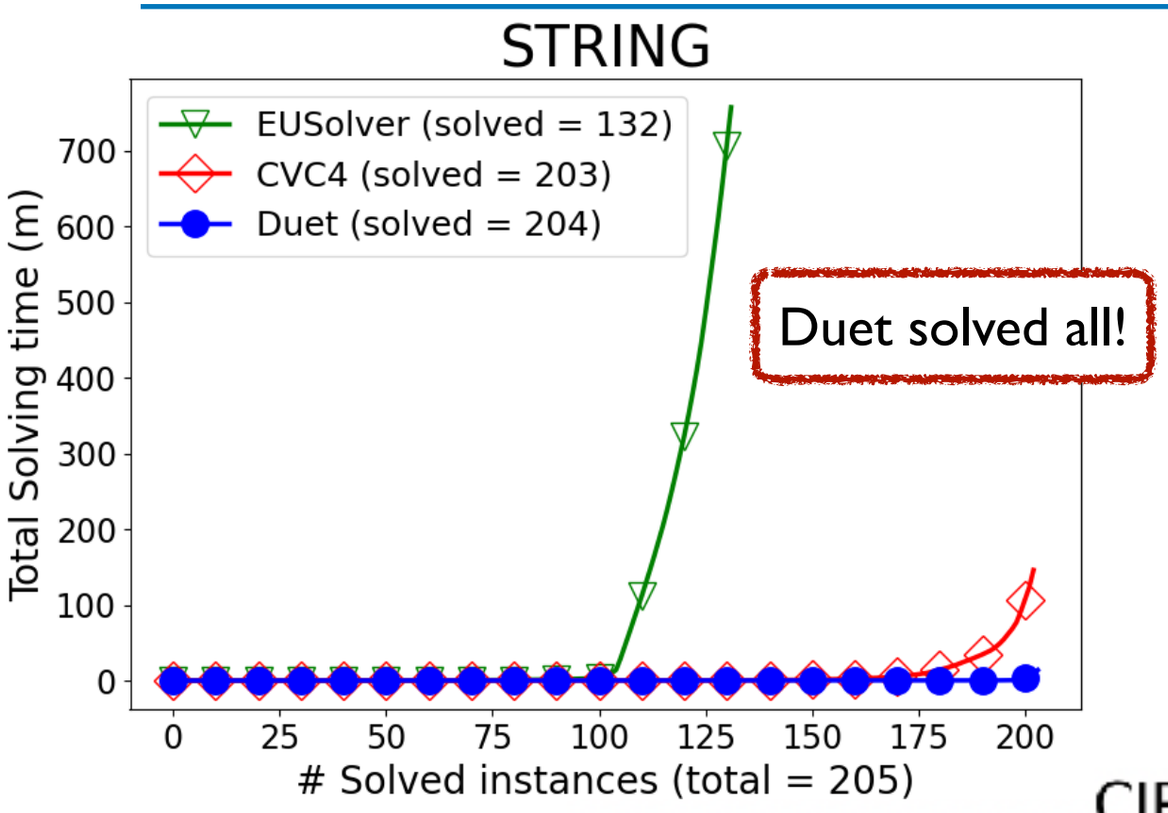
실험 결과

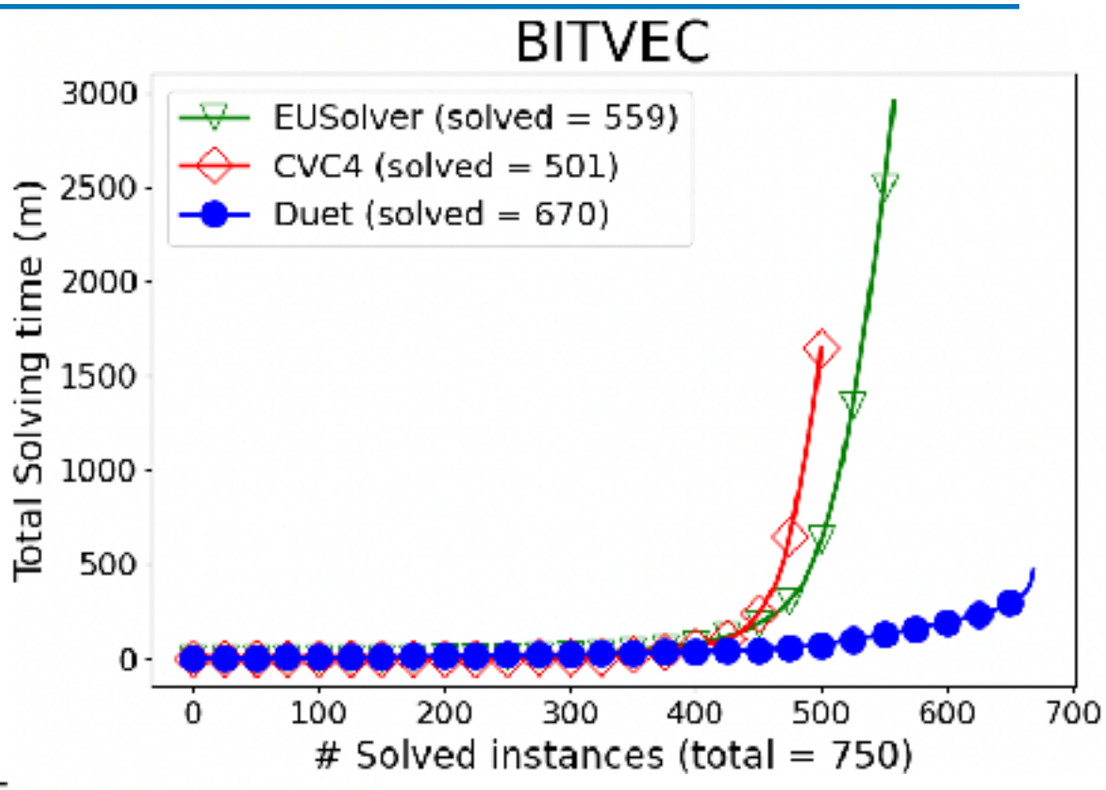

STRING
BITVEC
CIRCUIT
1443
1258
987
1500
1125
750
375
0
Solved Benchmarks
569
670
204
567
559
132
283
501
203
DUET
EUSolver
CVC4
Duet solved more problems than the others ...

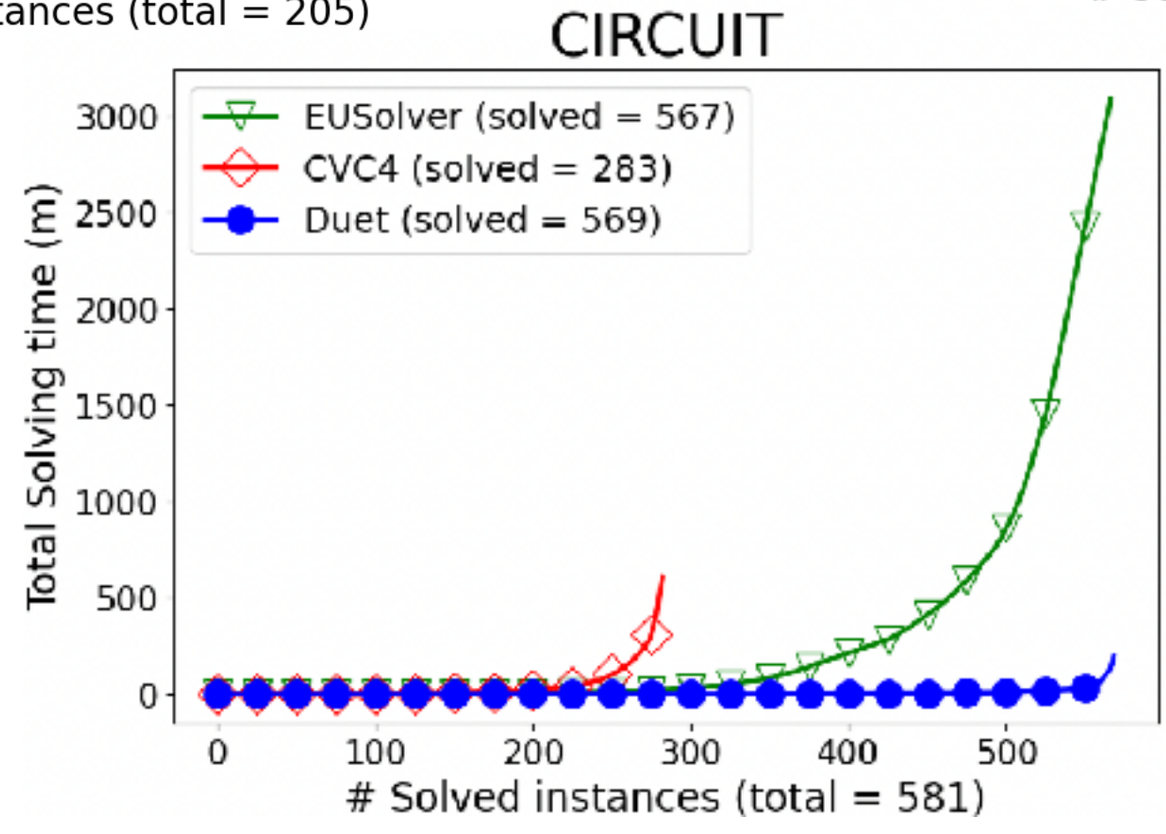
실험 결과


STRING
BITVEC
CIRCUIT
1030
101
339
1200
900
600
300
0
Fastest Solved Benchmarks
556
282
192
14
87
3
318
18
DUET
EUSolver
CVC4
... in the fastest speed!

실험 결과





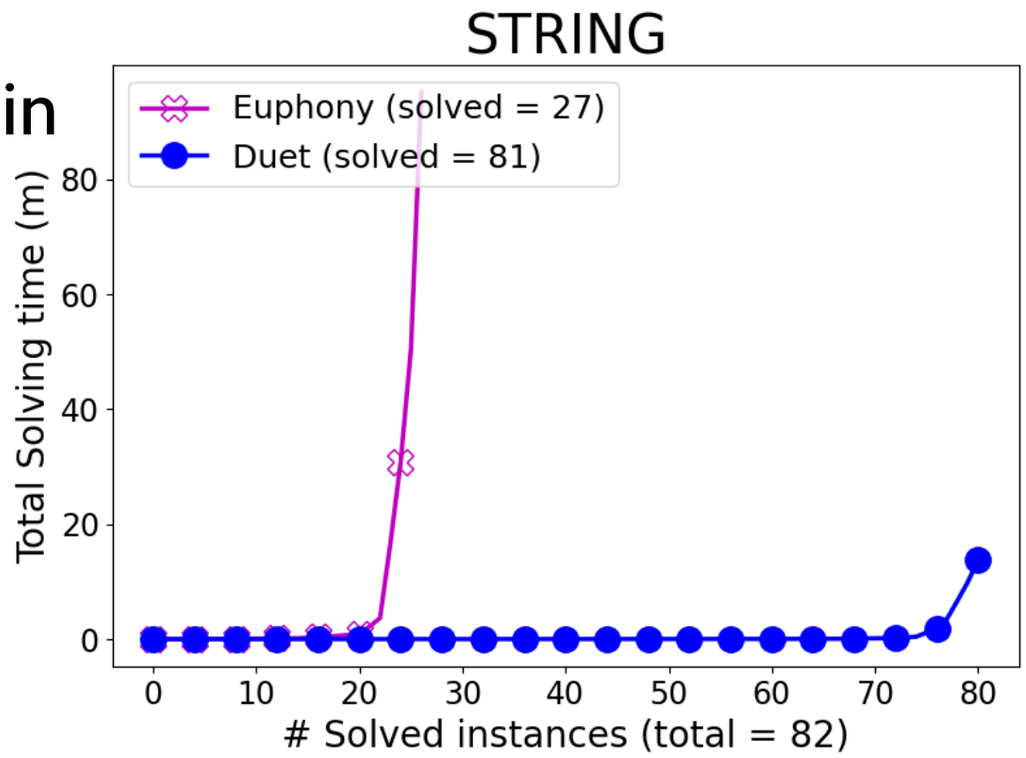


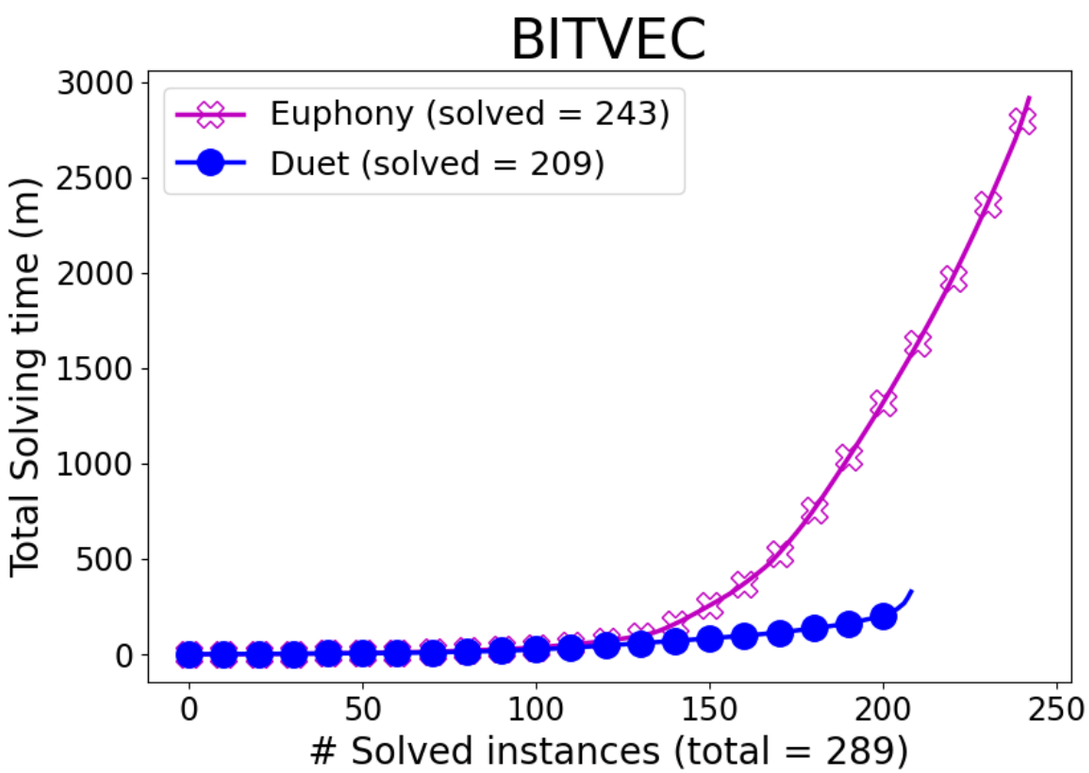
Duet solved

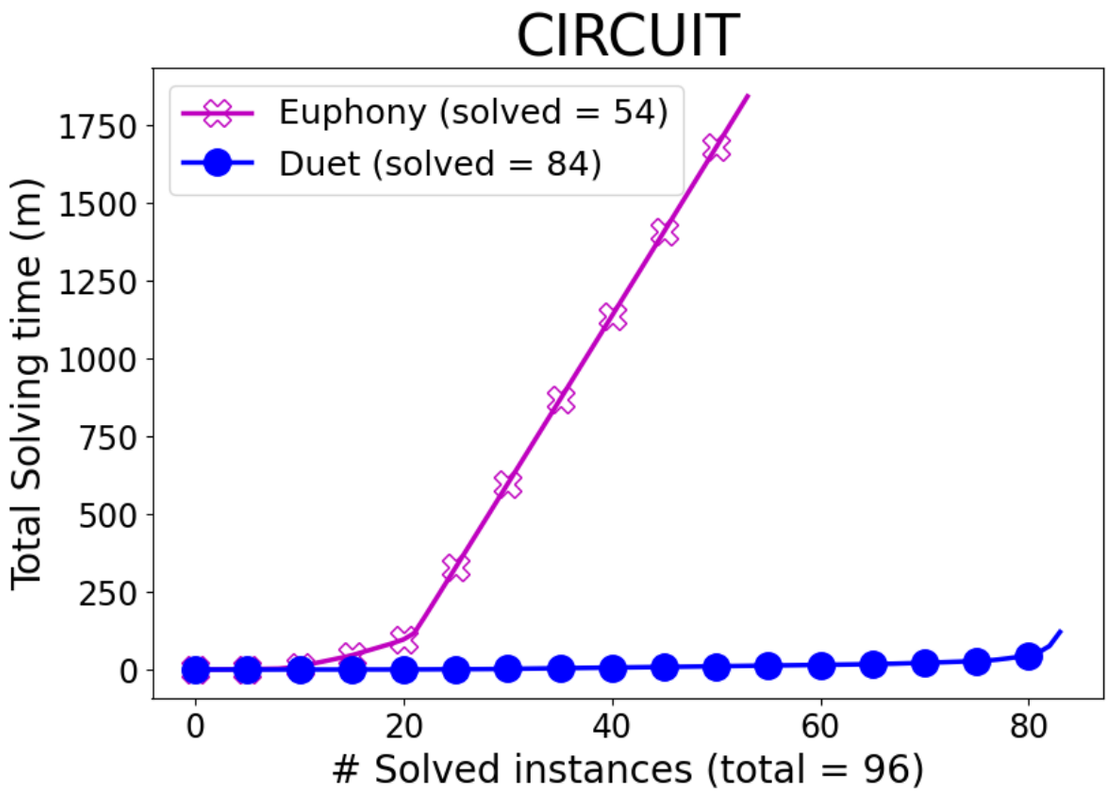
40% within **1 sec**
77% within **10 secs**
90% within **1 min.**

실험 결과 (vs. Euphony)

- Training: 1,069 solved EUSolver in 10 min
- Testing: 467
- # solved: Duet: 374, Euphony 324

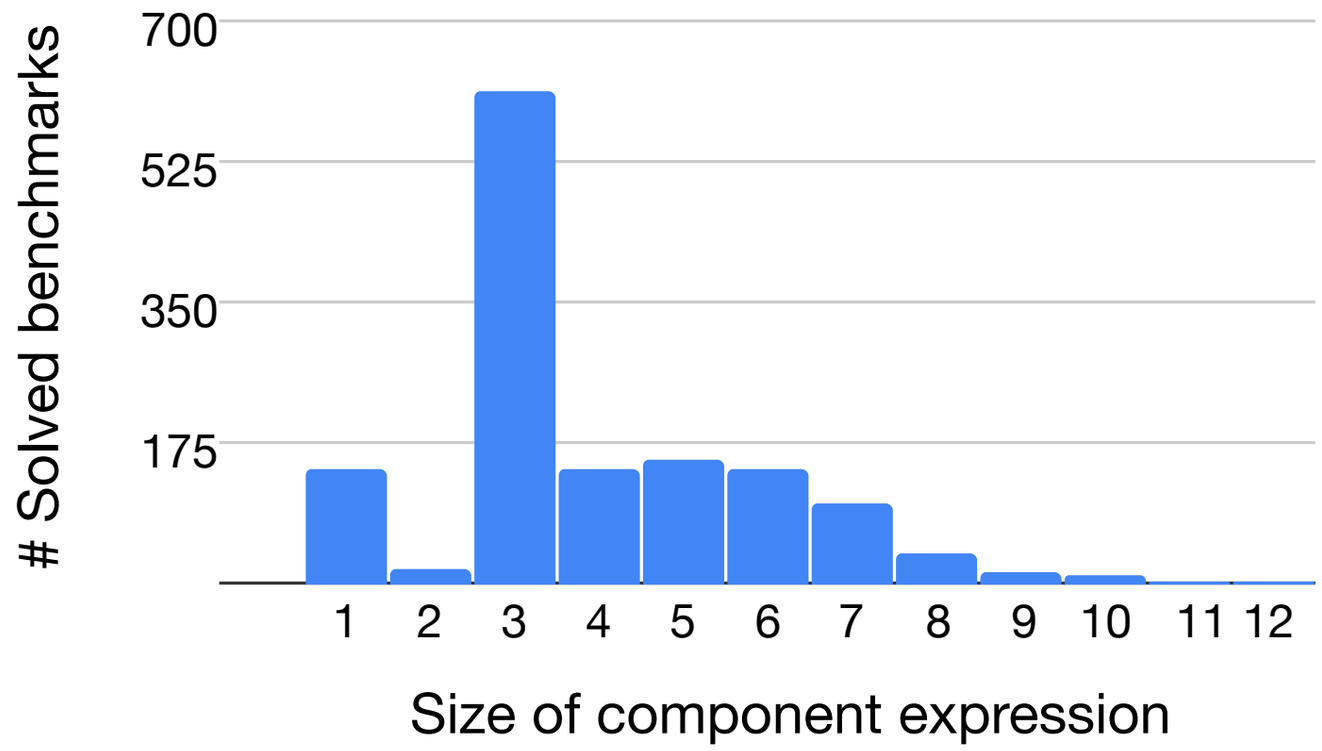






부품 크기 분석

- 해를 찾는데 필요한 부품 크기
  - STRING: 1 ~ 6 (avg: 1.8)
  - BITVEC: 3 ~ 5 (avg: 3.5)
  - CIRCUIT: 3 ~ 12 (avg: 5.9)



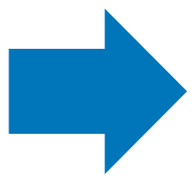
- 92% 의 문제가 크기 7 이하 부품으로 해결 가능.

내용

- 프로그램 합성 개괄
- 귀납 합성 전략
- **연역 합성 전략**

입출력 예제는 종종 (제약조건으로) 너무 느슨

```
1 # gcc -O3
2
3 movq rsi, r9
4 movl ecx, ecx
5 shrq 32, rsi
6 andl 0xffffffff, r9d
7 movq rcx, rax
8 movl edx, edx
9 imulq r9, rax
10 imulq rdx, r9
11 imulq rsi, rdx
12 imulq rsi, rcx
13 addq rdx, rax
14 jae .L0
15 movabsq 0x100000000, rdx
16 addq rdx, rcx
17 .L0:
18 movq rax, rsi
19 movq rax, rdx
20 shrq 32, rsi
21 salq 32, rdx
22 addq rsi, rcx
23 addq r9, rdx
24 adcq 0, rcx
25 addq r8, rdx
26 adcq 0, rcx
27 addq rdi, rdx
28 adcq 0, rcx
29 movq rcx, r8
30 movq rdx, rdi
```



```
1 # STOKE
2
3 shlq 32, rcx
4 movl edx, edx
5 xorq rdx, rcx
6 movq rcx, rax
7 mulq rsi
8 addq r8, rdi
9 adcq 0, rdx
10 addq rdi, rax
11 adcq 0, rdx
12 movq rdx, r8
13 movq rax, rdi
```

Montgomery multiplication kernel from the OpenSSL RSA library. Compilations shown for gcc –O3 (left) and a stochastic optimizer (right).

http://stoke.stanford.edu

입출력 예제는 종종 (제약조건으로) 너무 느슨

- 원본 프로그램 f 과 의미가 같으면서 최적화된 프로그램 f' 을 원할 경우,
  - 귀납합성으로 간단히 생각 할 수 있는 방법: 랜덤 입력들을 생성하여 원본 프로그램에 대해서 돌린 후 출력값 얻은 후, 입출력 예제들을 사용
  - 문제: 올바름 조건 100% 보장 안됨
- 제대로 된 올바름 조건: $\forall x.\ f(x) = f'(x)$

입출력 예제는 종종 (제약조건으로) 너무 느슨

- 올바름 조건이 논리식으로 사전/사후(pre/post) 조건으로 기술된 경우에도 기존 귀납 합성 전략 먹통.

```
int W = 32;

void swap(ref bit[W] x, ref bit[W] y){
    if(??){ x = x ^ y;}else{ y = x ^ y; }
    if(??){ x = x ^ y;}else{ y = x ^ y; }
}

harness void main(bit[W] x, bit[W] y){
    bit[W] tx = x; bit[W] ty = y;
    swap(x, y);
    assert x==ty && y == tx;
}
```

- Program sketch: https://people.csail.mit.edu/asolar/sketch-1.7.6.tar.gz

프로그램 합성 문제의 일반적 기술

$$\exists f.\forall \mathbf{x}.P(f,\mathbf{x})$$

There exists a function `f` such that for all **x**, property `P` holds

- **모든 가능한** 입력에 대해 (∀**x**) 어떤 성질을 만족하는 함수 찾기

- 기존 SMT solvers로 풀 수 있는가? ⇒ **NO!** 2차 변수 `f` 와 ∀(universal quantifier) 이 붙은 1차 변수가 있는 식 잘 못 다룸.

프로그램 합성 문제의 일반적 기술

$$\exists \mathtt{f}.\forall \mathbf{x}.\mathtt{P}(\mathtt{f},\mathbf{x})$$

There exists a function `f` such that for all **x**, property `P` holds

- 그냥 가능한 `f`를 나열하면서 찾아보기? ⇒ 2차 변수가 사라져서 SMT solver가 다룰 수 있으나, SMT solver 호출(계산비용 비쌈) 횟수가 너무 많아짐.

Counterexample guided Inductive Synthesis (CEGIS)

- 2008년 Armando Solar-Lezama 가 제안한 알고리즘.
- 기존의 귀납 합성 전략을 일반적인 제약조건을 다루는 연역 합성에 사용할 수 있게 해 줌.
- 합성 대상 함수의 후보를 제안하는 탐색파트(generator)와, 제안된 후보가 정말 맞는지 체크하는 검증파트(verifier)로 나뉨.

Example

- Goal: a function `f` computes the maximum of integers `x` and `y`
-  Specification

  Syntactic specification:

  if C then S else S

  $$S \to x \mid y \mid n \mid ite(C, S, S)$$

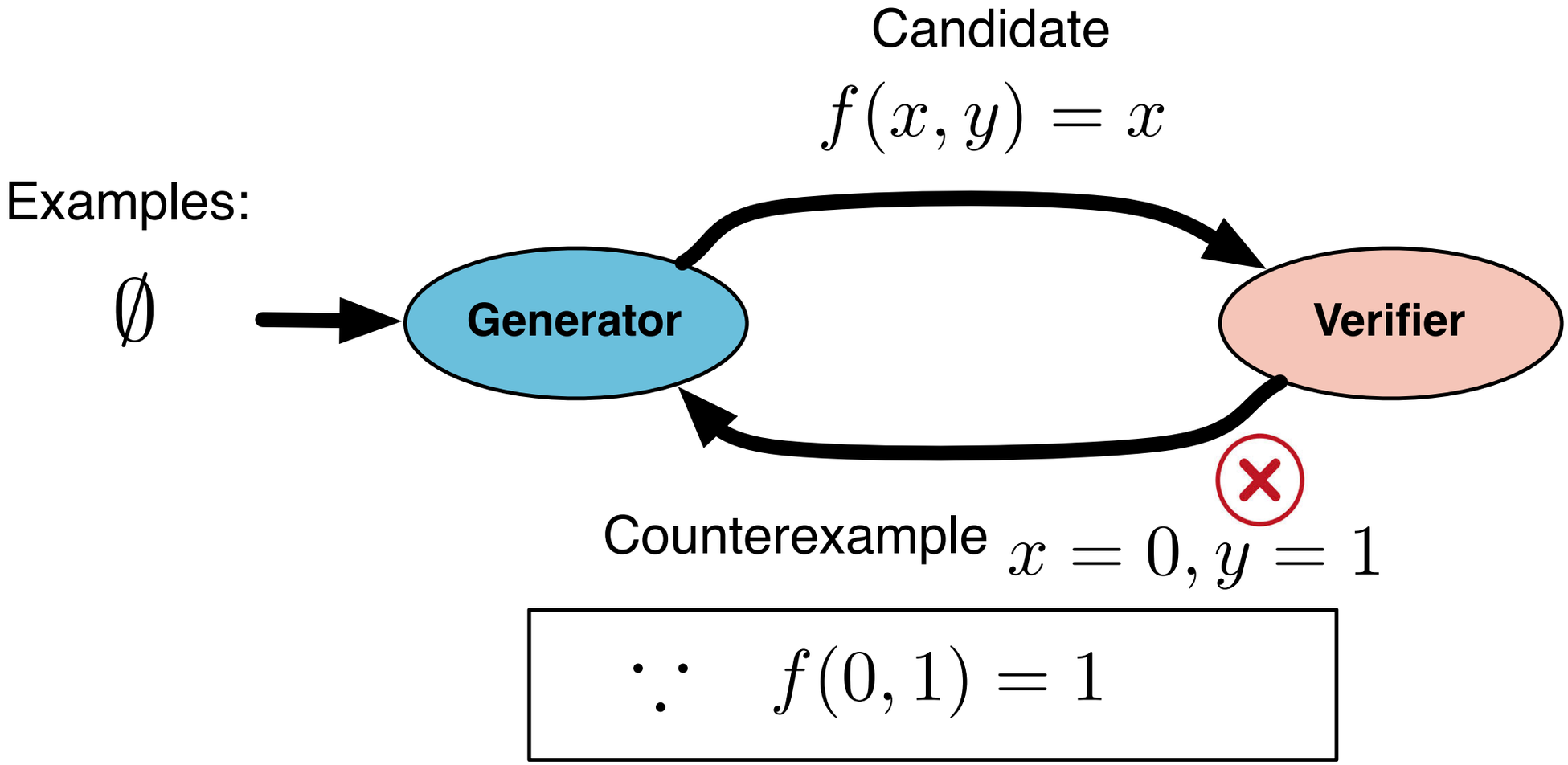
  $$C \to S \leq S \mid \neg C \mid C \& C$$

  Semantic specification:

  $$\forall x, y.\ f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$$

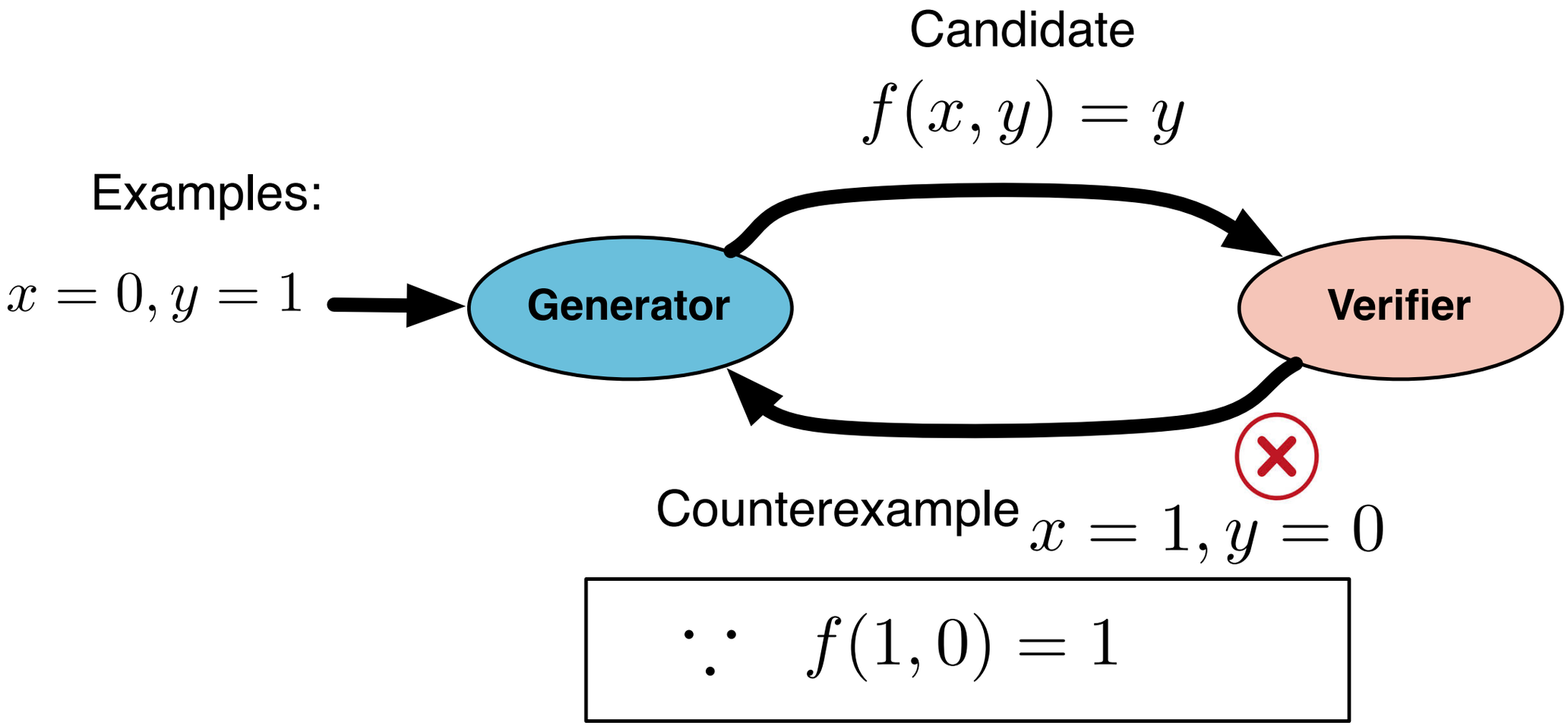
-  Solution $f(x, y) = ite(x > y, x, y)$

CEGIS + 상향식 나열



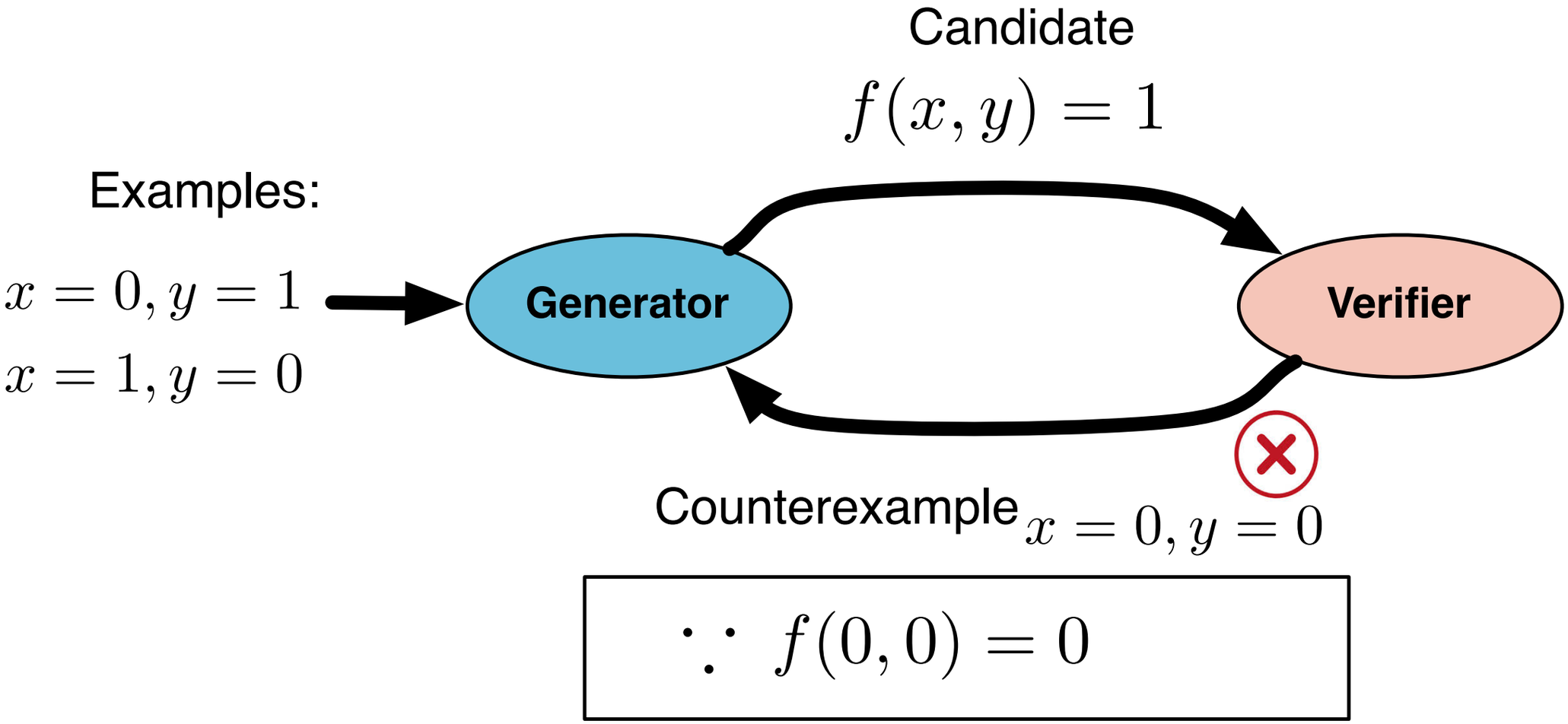
$$\because \quad f(0, 1) = 1$$

CEGIS + 상향식 나열



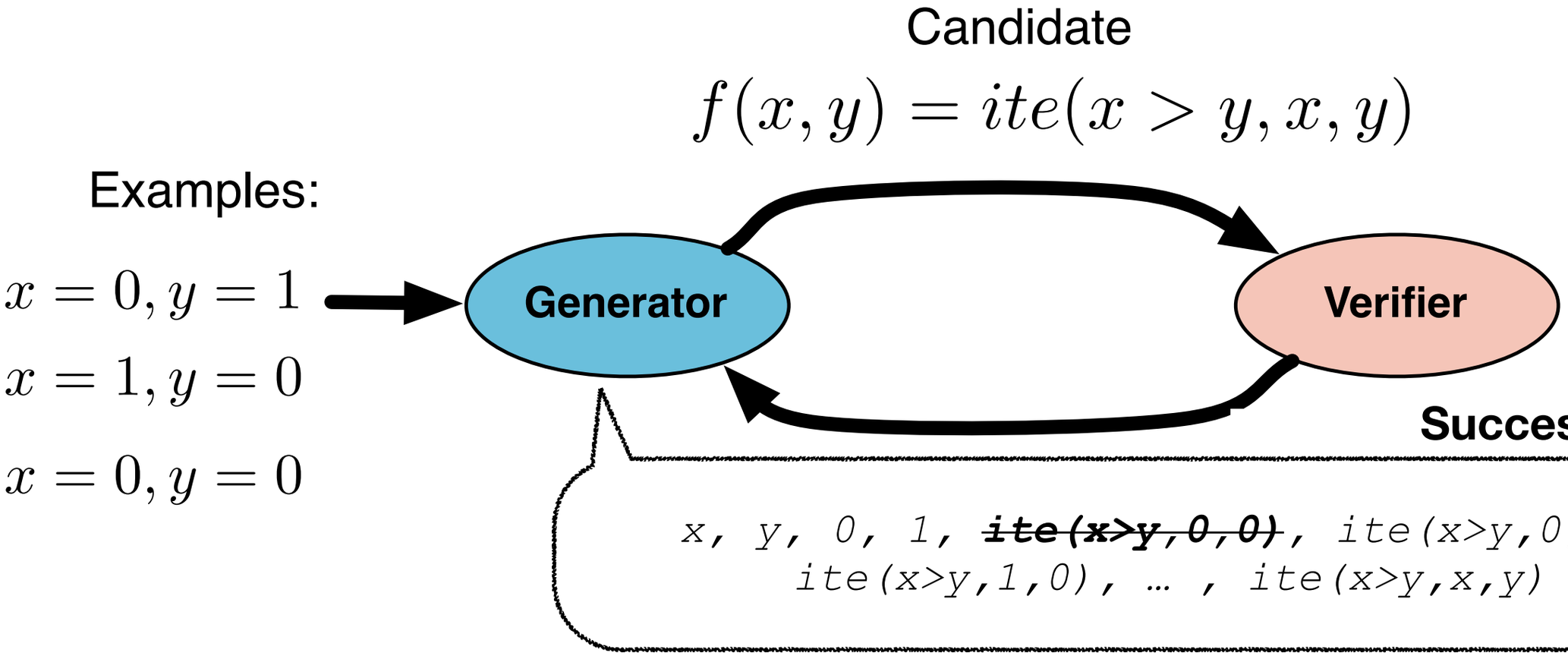
$$\because \quad f(1, 0) = 1$$

CEGIS + 상향식 나열



$$\because \ f(0,0) = 0$$

CEGIS + 상향식 나열



최적화: *ite(x>y,0,0)* **는 무시됨.**

$\because\ ite(x > y, 0, 0) \approx 0$ wrt examples

CEGIS가 가져다주는 이점

- 검증기와 탐색기는 각기 따로따로 설계되어도 됨 (탐색기는 임의의 귀납 합성전략, 검증기는 임의의 검증 방식 ...)

- 검증기는 2차변수 없이 1차변수만 다루어도 됨.

- 탐색기는 2차변수를 다루지만 universal quantifier 는 다룰 필요가 없음.

정리

- 프로그램 합성 문제 = 제약조건 + 탐색공간
- 가능 탐색 전략은 제약조건, 탐색공간에 따라 결정됨.
- 제약조건이 입출력예제인 경우 귀납합성전략 사용
- 제약조건이 논리식인 경우 귀납합성전략 + CEGIS 사용