통계 모델을 활용한 효과적인 프로그램 자동 생성

(Effective Program Generation using Learned Probabilistic Models)

이 우 석

한양대학교 소프트웨어학부

한국정보과학회 KCC2019

신진교수 최신연구소개 특별세션

6월 27일, 제주 ICC

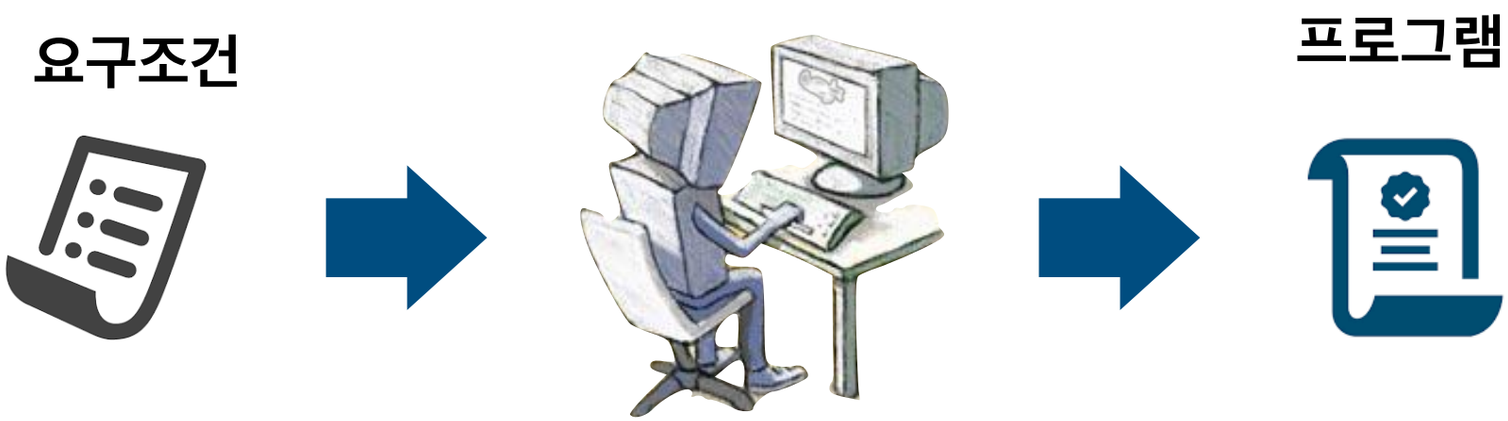
발표자 소개

이 우 석

- 한양대학교 ERICA 소프트웨어학부 조교수
- University of Pennsylvania 박사 후 연구원
  (지도교수: Mayur Naik)
- 서울대학교 컴퓨터공학 박사
  (지도교수: 이광근)
- 연구분야: 프로그램 합성, 프로그램 분석

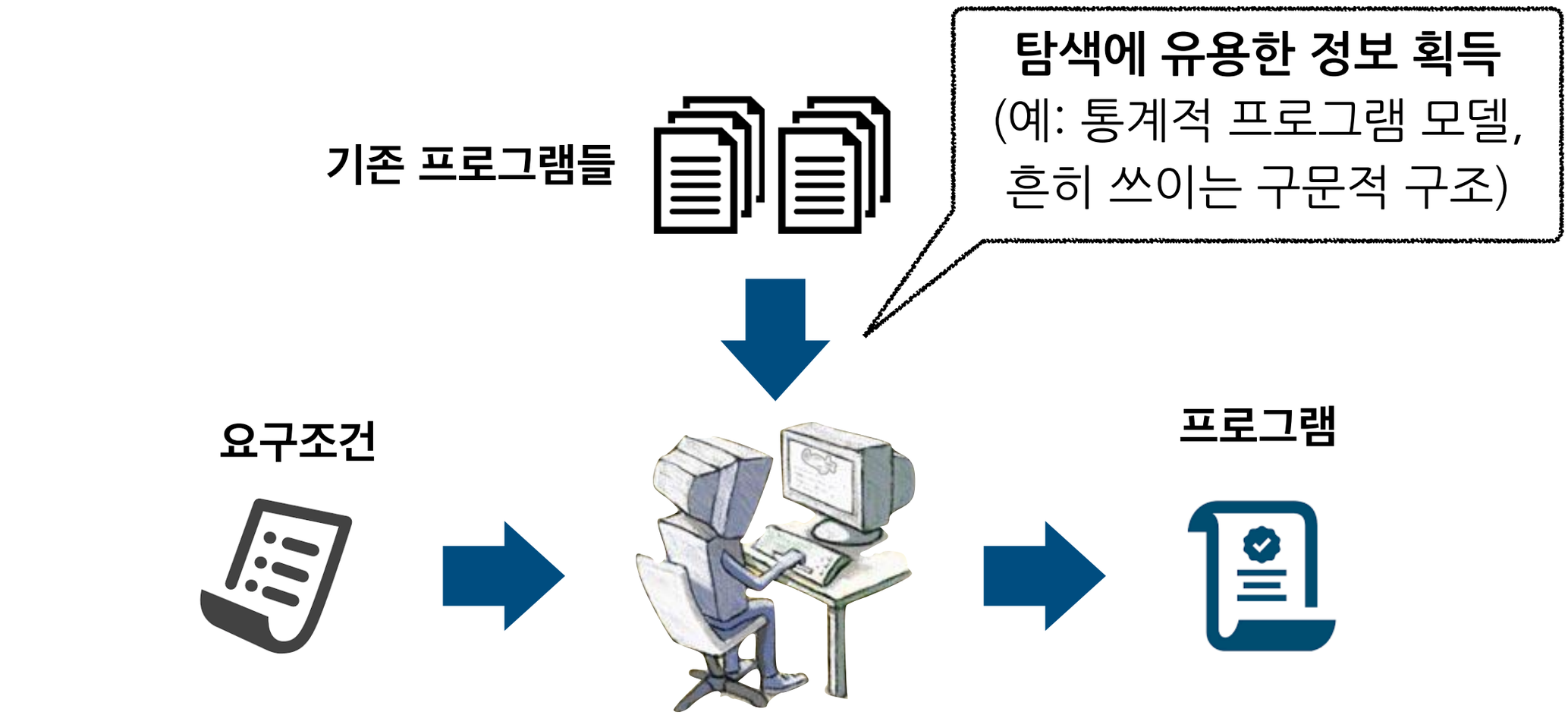
프로그램 합성 Program Synthesis

- 사용자가 원하는 요구조건을 만족시키는 프로그램을 자동으로 생성



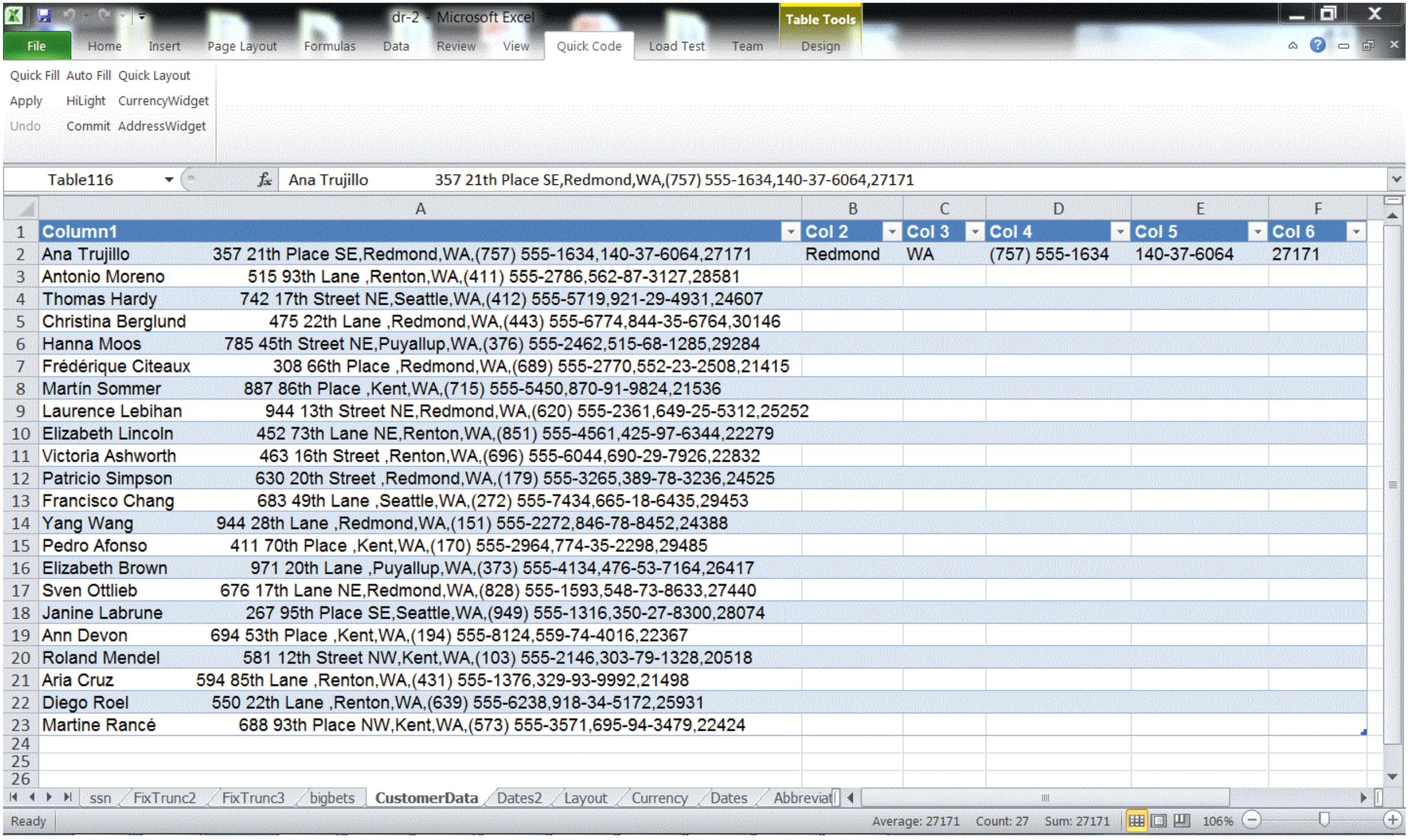
- 요구조건: 프로그래밍 지식 없이 쉽게 작성하고 이해할 수 있는 형태로 기술
  - 원하는 프로그램의 입력/출력 예제
  - 원하는 프로그램이 만족시켜야할 논리식

연구 소개 요약

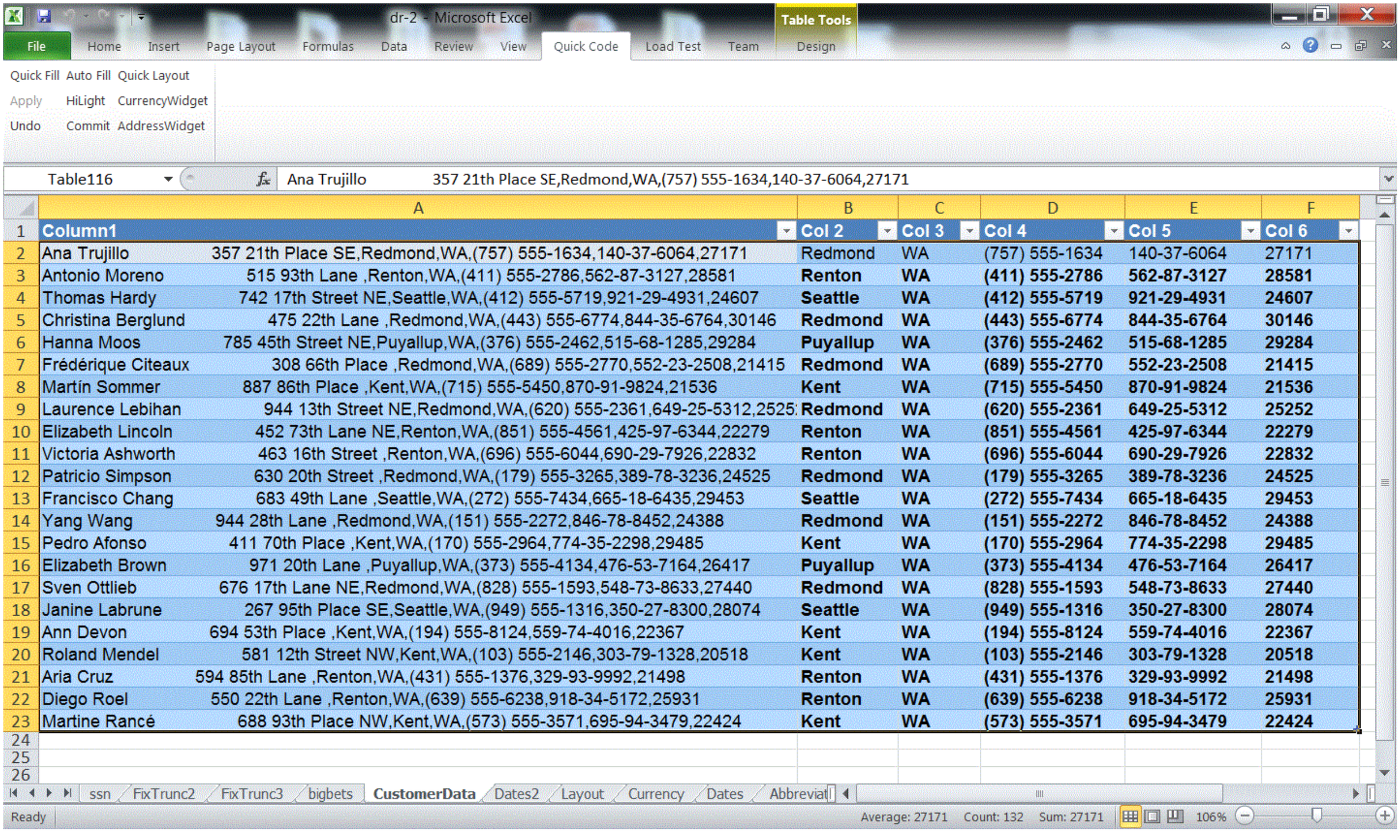

기존 프로그램들
탐색에 유용한 정보 획득
(예: 통계적 프로그램 모델,
흔히 쓰이는 구문적 구조)
요구조건
프로그램
탐색 성능 가속화
• 통계적 프로그램 모델을 이용한 프로그램 합성 가속화
• 강화학습을 이용한 프로그램 축소 가속화
• 학습된 템플릿을 이용한 Datalog 프로그램 합성 가속화

프로그램 합성의 성공사례: FlashFill

프로그램 합성의 성공사례: FlashFill



프로그램 합성의 성공사례: FlashFill



프로그램 합성의 사례: 프로그램 최적화

합성 목표: 주어진 프로그램과 의미가 같으면서 더 효율적인 프로그램 찾기

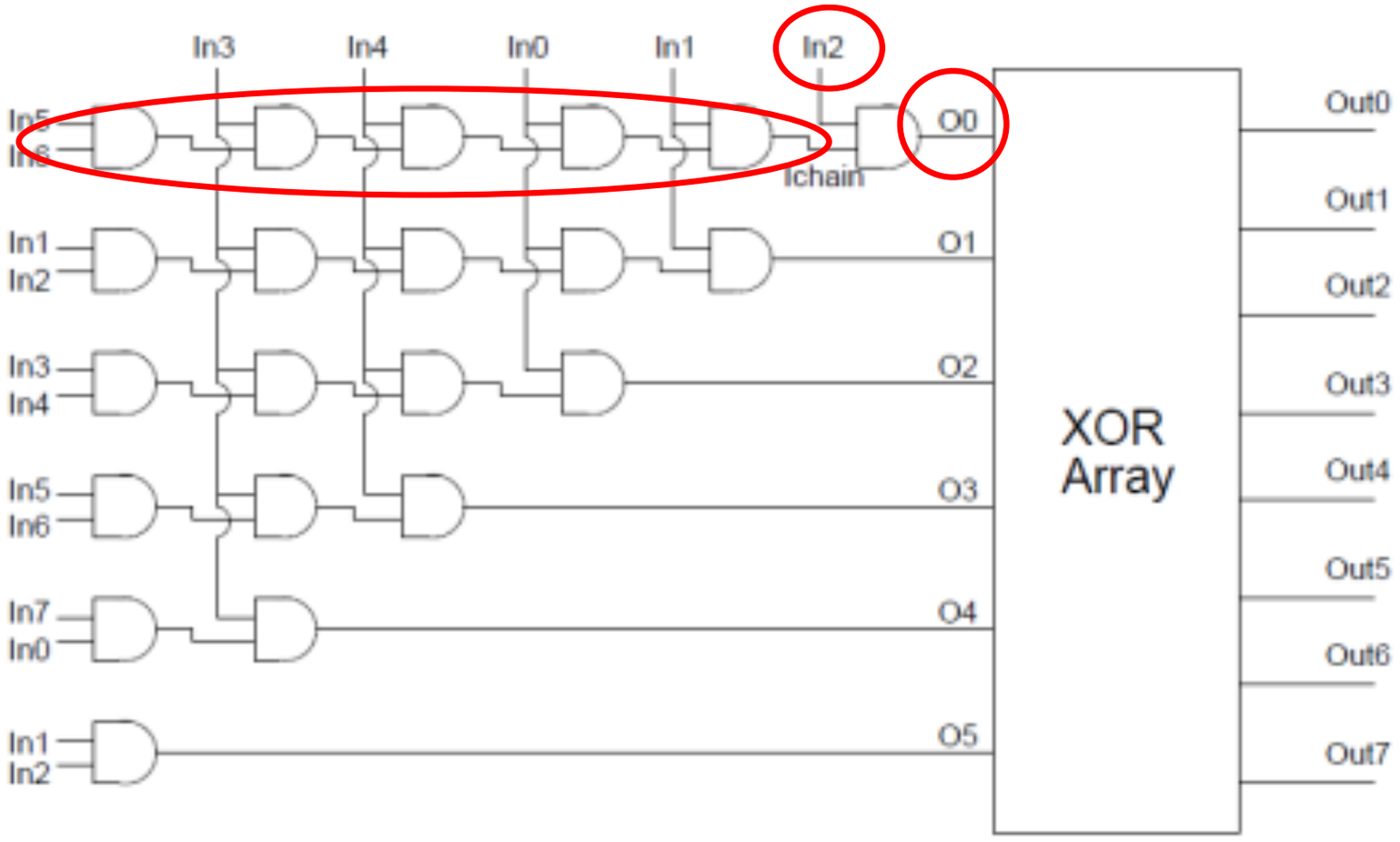
32비트 정수 2개의
평균 계산

```
average (bitvec[32] x, y) {
      bitvec[64] x1 = x;
      bitvec[64] y1 = y;
      bitvec[64] z1 = (x1+y1)/2;
      bitvec[32] z = z1;
      return z;
}
```

64비트값 사용 않고
동일한 코드 작성
(비트연산들만 사용해서)

```
average (x, y) =
  (x and y) + [(x xor y) shift-right 1 ]
```

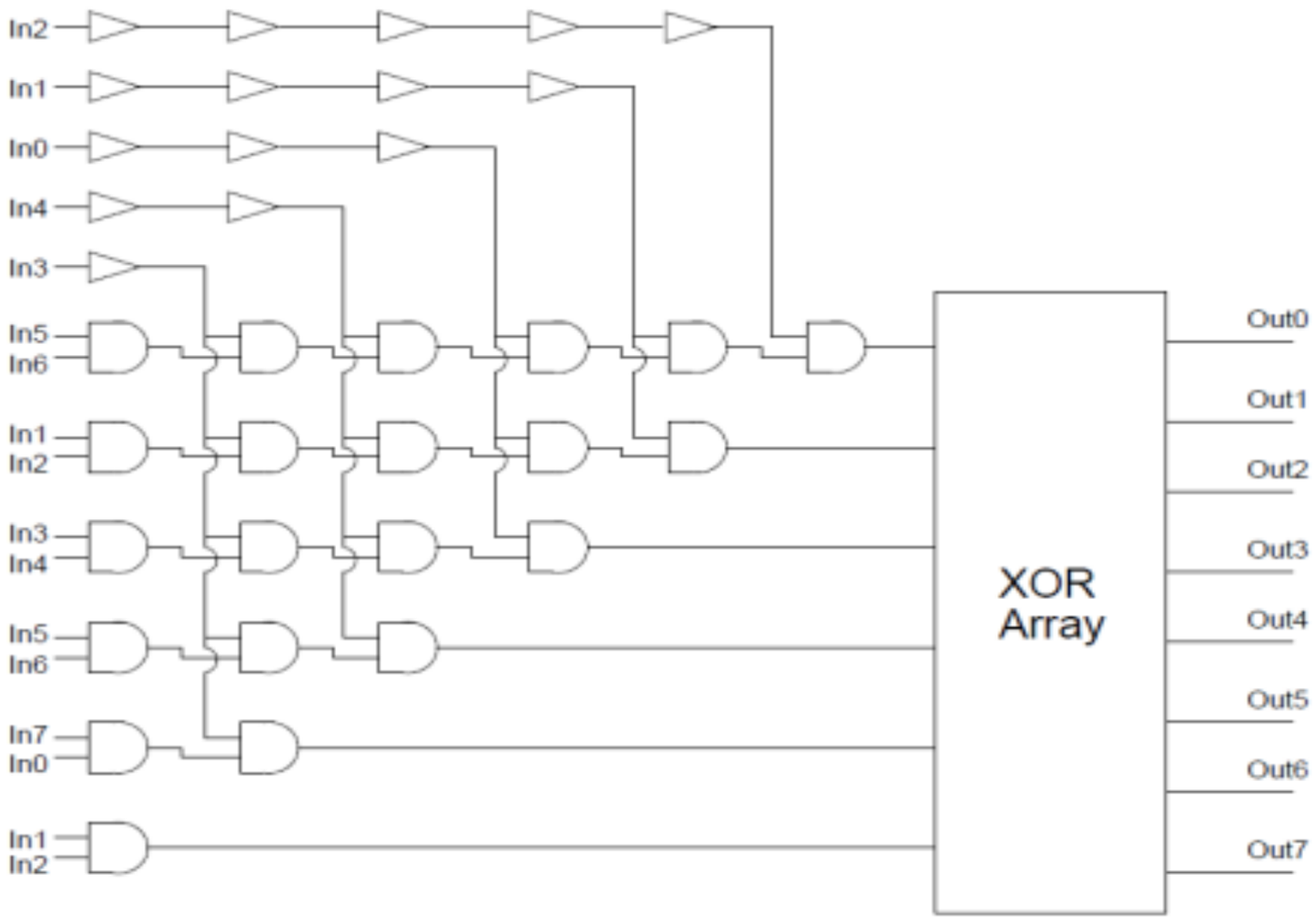
프로그램 합성의 사례: 부채널 공격side-channel attack에 대한 방어



PPRM1 AES S-Box 구현 [Morioka and Satoh, 2002]

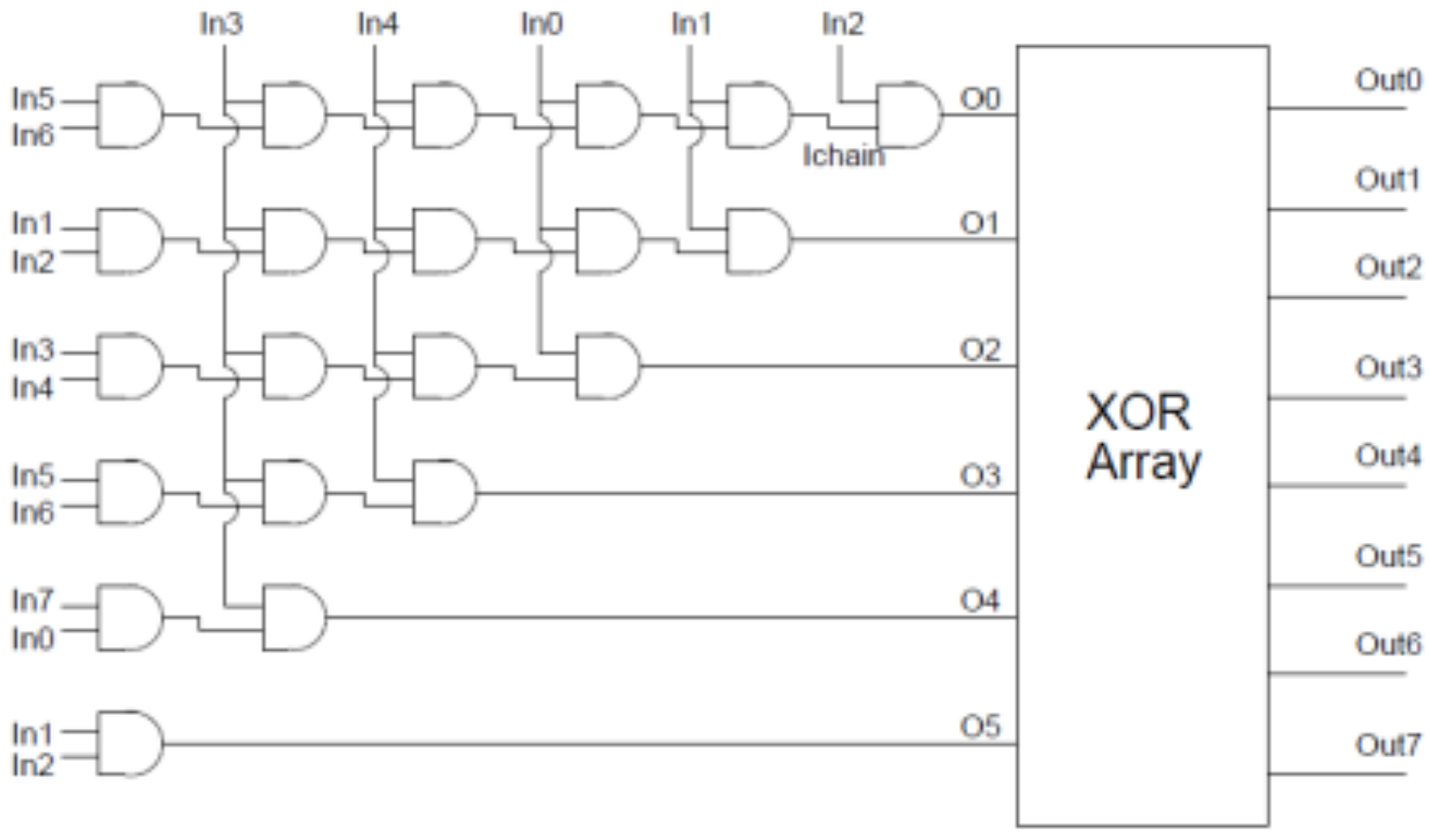
취약점: 시간 차 공격Timing-based attack 으로 민감한 입력값 In2에 대한 정보 획득

프로그램 합성의 사례: 부채널 공격side-channel attack에 대한 방어



부채널 공격에 대비한 회로: 모든 입력-to-출력 경로가 같은 횟수의 연산을 거치도록 수동 작성된 회로 [Schaumont et al. DATE 2014]

프로그램 합성의 사례: 부채널 공격side-channel attack에 대한 방어



합성 목표: 주어진 회로 C에 대해 다음 조건을 만족하는 회로 C'찾기

1. C' 는 C 와 하는일이 동일 [프로그램 의미에 대한 조건]
2. C' 의 모든 입력-출력 경로는 같은 길이를 갖는다 [프로그램 생김새에 대한 조건]

기존의 회로 최적화 도구 (EDA)는 위 조건을 만족하는 회로 C' 생성 불가

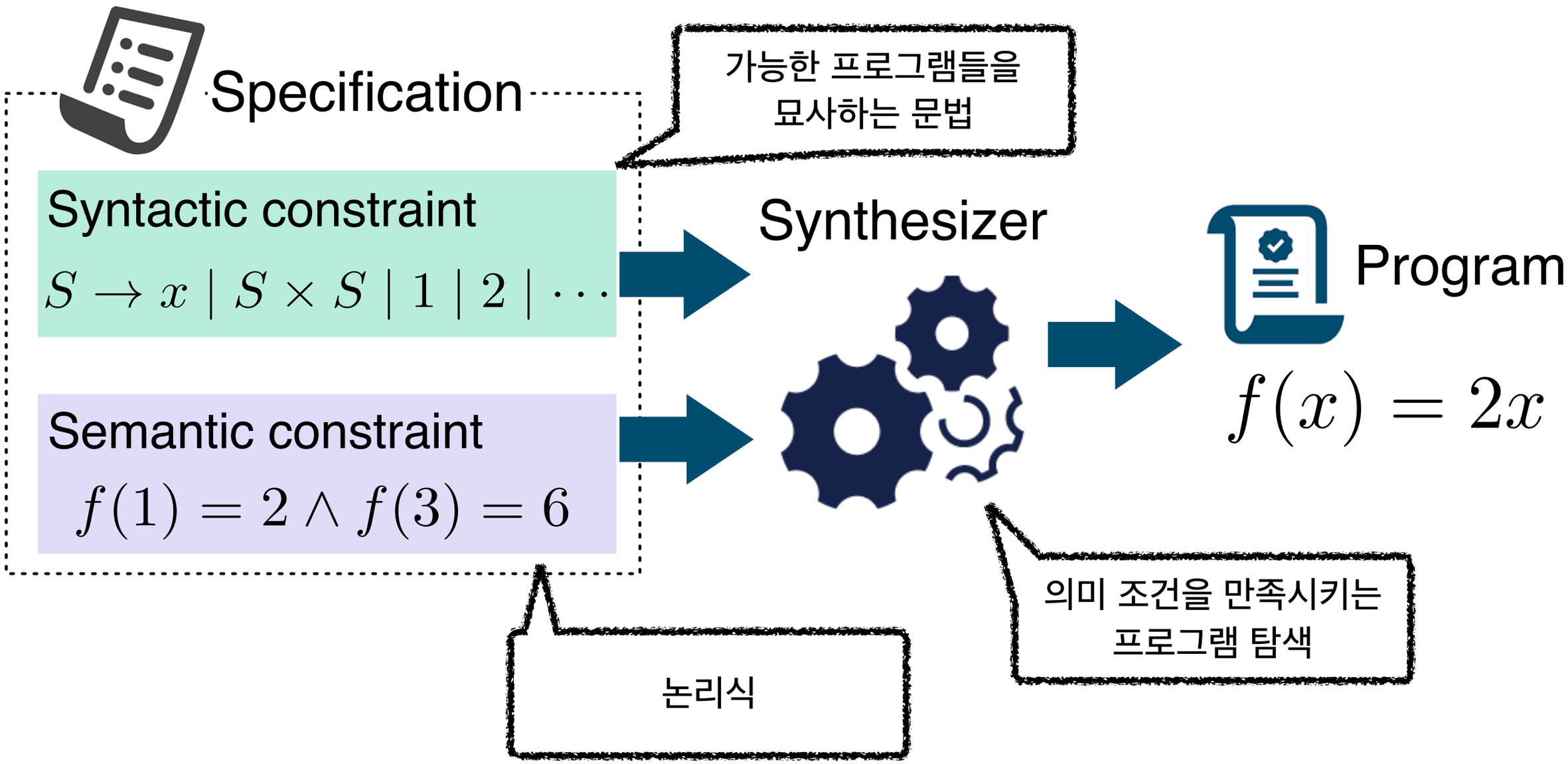
프로그램 합성 문제 표준 포맷: SyGuS
(**Sy**ntax-**gu**ided Program **S**ynthesis)

- 다양한 응용분야에서의 프로그램 합성 문제들
  - 입출력 예제를 만족시키는 프로그램 찾기
  - 프로그램 최적화
  - 자동 프로그램 수정
  - 입출력 테이블로부터 SQL 쿼리 생성
  - 프로그램 검증을 위한 불변식invariant 추출
- **위 문제들의 핵심:**

  원하는 프로그램의 생김새에 대한 조건syntactic constraint과
  의미에 대한 조건semantic constraint 이 주어졌을 때, 이를 만족하는 프로그램 찾기

프로그램 합성 문제 표준 포맷: SyGuS
(**Sy**ntax-**gu**ided Program **S**ynthesis)

프로그램 합성 문제를 형식화 / 표준화



†http://www.sygus.org

기존의 프로그램 합성 전략들

- **나열식**: 모든 가능한 후보나열 (가지치기pruning 최적화도 함께)
  - EUSolver: Udupa et al. (PLDI'13, TACAS'17)
- **기호식:** 문제를 논리식의 해를 찾는 문제로 환원constraint solving
  - CVC4: Reynolds et al. (CAV'15, CAV'18, IJCAR'18)
- **확률적 방식:** 프로그램 랜덤 변환probabilistic walk
  - STOKE: Schkufza et al. (ASPLOS'13, ASPLOS'17)

기존 기술들의 한계:
그럴싸한 프로그램을 먼저 찾는 쪽으로
탐색과정이 안내되지 않음

프로그램의 통계적 규칙성 Statistical regularity

- "프로그램은 반복적이고 예측가능한 패턴들로 이루어져 있다." [Hindle et al. ICSE'12]

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

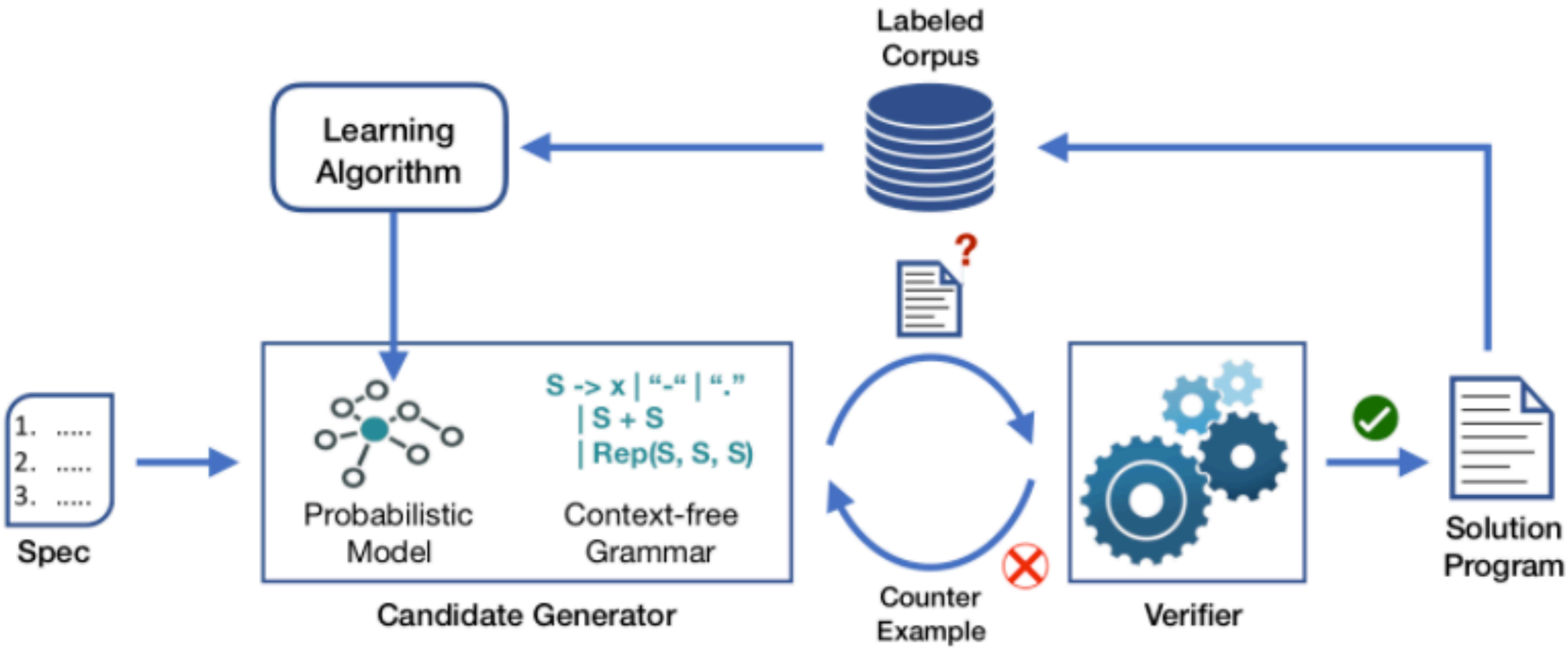
- 통계적 프로그램 모델 Statistical program models : 프로그램들의 확률 분포 정의

$$Pr(?? \rightarrow \texttt{i++} \mid \texttt{for (i = 0; i < 100; ??)}) = 0.80$$
$$Pr(?? \rightarrow \texttt{i--} \mid \texttt{for (i = 0; i < 100; ??)}) = 0.01$$

  - 예) n-gram, neural network, probabilistic context-free grammar (PCFG), …

- 다양한 응용 예: 자동 코드 완성code completion, 역난독화deobfuscation, 프로그램 자동 수정program repair, 등등.

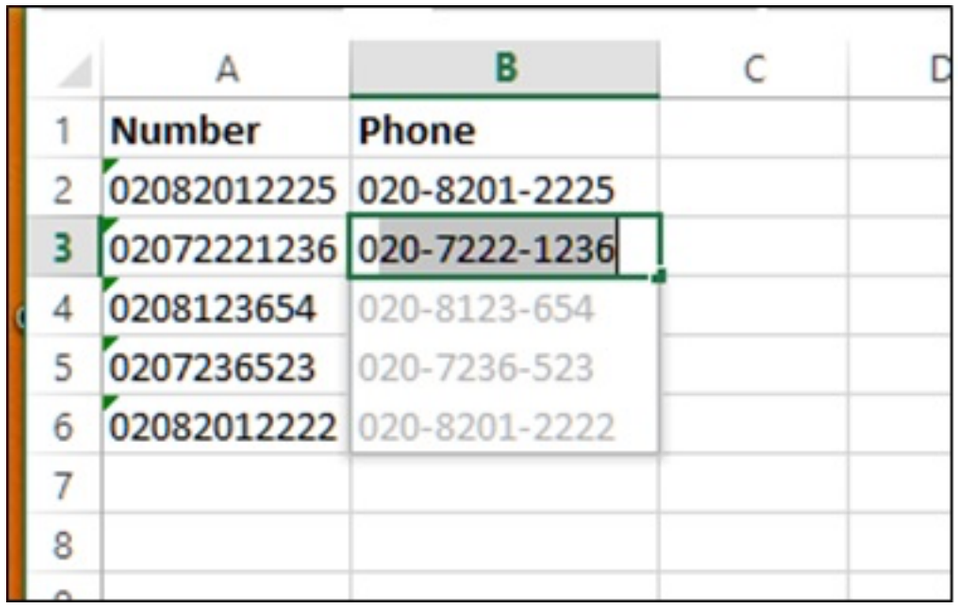
Euphony: 데이터 기반data-driven 프로그램 합성기†



Accelerating Search-Based Program Synthesis Using Learned Probabilistic Models. ACM Conference on Programming Language Design and Implementation (***PLDI***), 2018. (프로그래밍 언어분야 최우수 국제학회)

† http://github.com/wslee/euphony

기존 방법 대비 뛰어난 성능



	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				

예제로부터 문자열 변환 프로그램 찾기

205 문제

complement

```
~ 01010001110101110000000000001111
  10101110001010001111111111110000
```

bitwise and

```
  01010001110101110000000000001111
& 00110001011011100011000101101110
  00010001010001100000000000001110
```

bitwise or

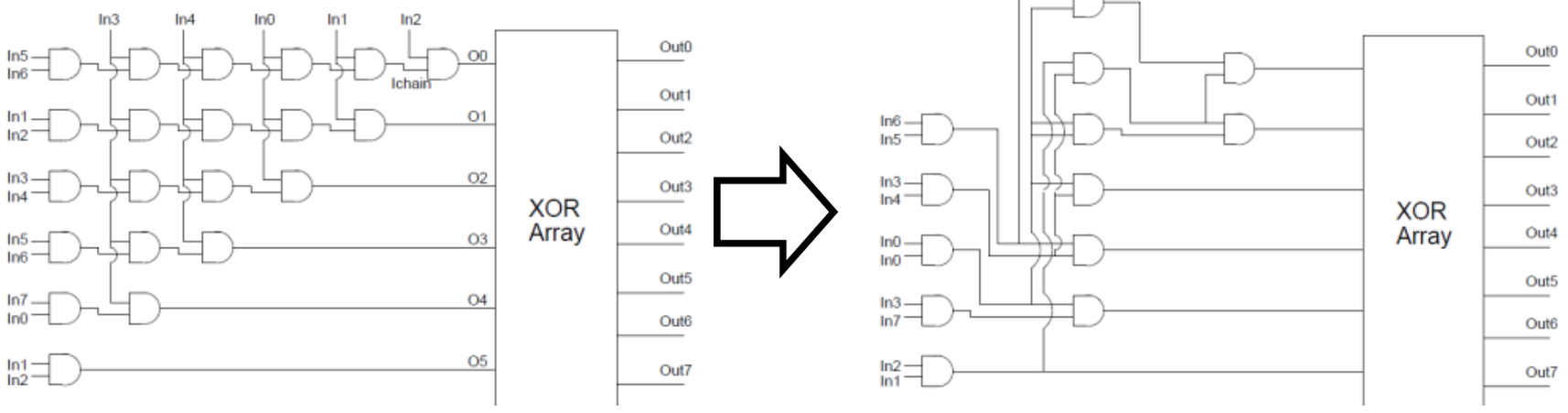
```
  01010001110101110000000000001111
| 00110001011011100011000101101110
  01110001111111110011000101101111
```

bitwise xor

```
  01010001110101110000000000001111
^ 00110001011011100011000101101110
  01100000101110010011000101100001
```

효율적인 비트벡터 알고리즘 찾기

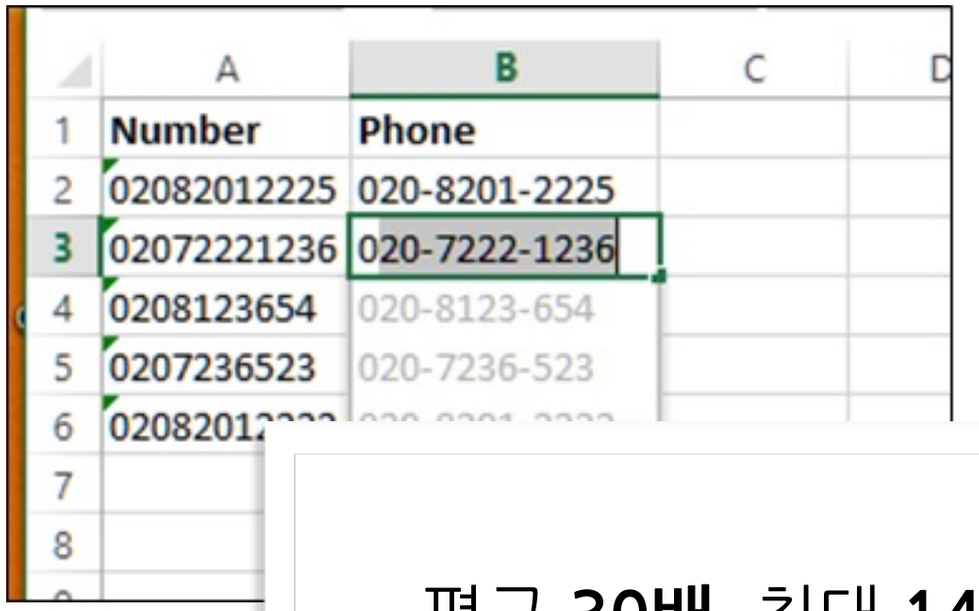
750 문제



부채널 공격에 안전한 회로 찾기

212 문제

기존 방법 대비 뛰어난 성능



complement

```
~ 01010001110101110000000000001111
  10101110001010001111111111110000
```

bitwise and

```
  01010001110101110000000000001111
& 00110001011011100011000101101110
  00010001010001100000000000001110
```

bitwise or

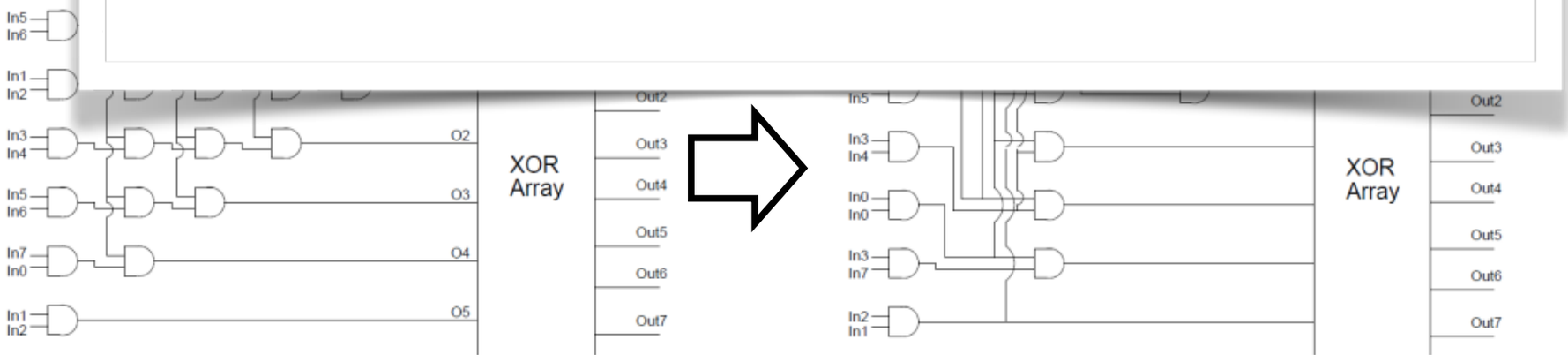
```
  01010001110101110000000000001111
| 00110001011011100011000101101110
  01110001111111110011000101101111
```

```
0001111
1101110
1100001
```

평균 **30배**, 최대 **140배** 빠른 성능 (vs. FlashFill)

평균 **15배**, 최대 **400배** 빠른 성능 (vs. EUSolver)

예제로부터 **리즘 찾기**



부채널 공격에 안전한 회로 찾기

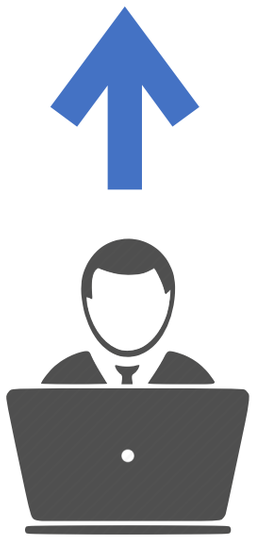
212 문제

프로그램 축소 Program reduction

tar 원래 버전

- Out-of-the-box Linux
- 97 커맨드라인 옵션
- 45,778 줄
- 13,227 명령문
- 보안 취약점: CVE-2016-6321

Manual



tar 경량화 버전

- BusyBox Utility Package*
- 8 커맨드라인 옵션
- 3,287 줄
- 403 명령문
- 알려진 취약점 없음

*https://busybox.net

자동 프로그램 축소

tar 원래 버전

- Out-of-the-box Linux
- 97 커맨드라인 옵션
- 45,778 줄
- 13,227 명령문
- 보안 취약점: CVE-2016-6321

Automatic

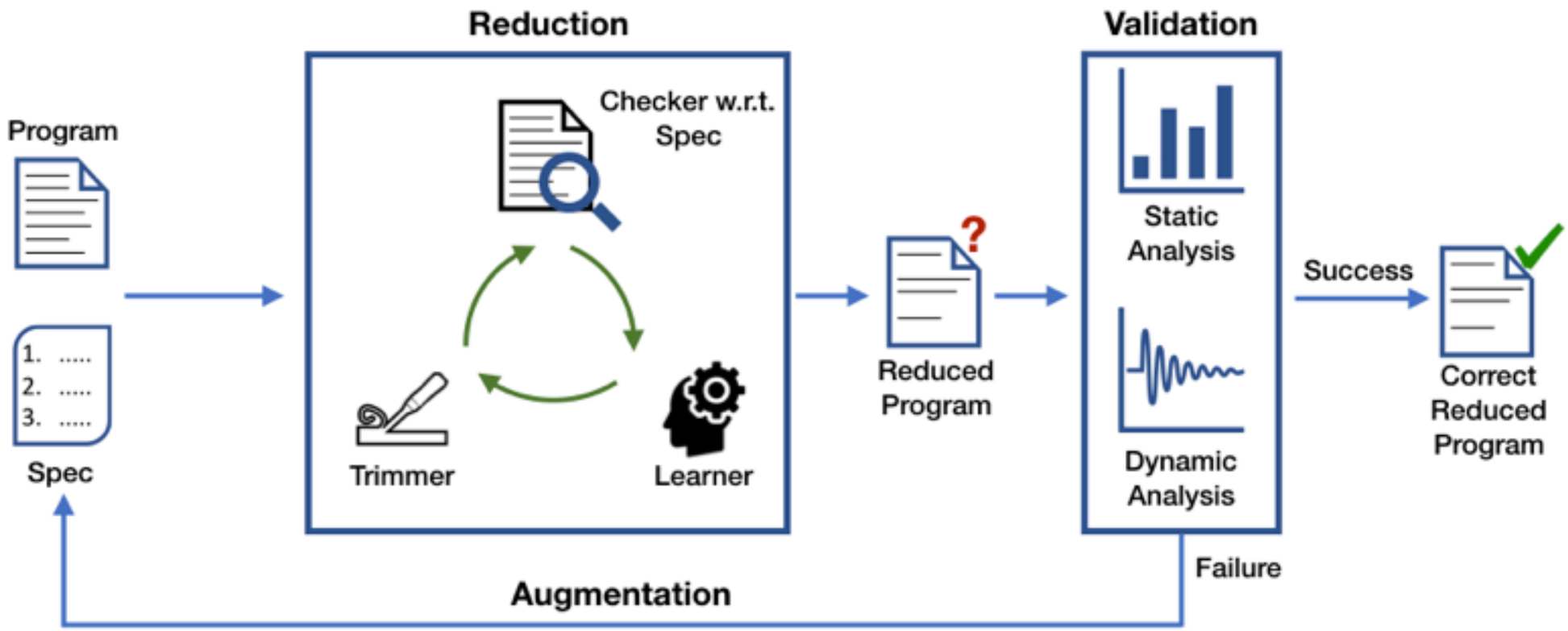
요구조건

tar 경량화 버전

- BusyBox Utility Package*
- 8 커맨드라인 옵션
- **1,646** ~~3,287~~ 줄
- **518** ~~403~~ 명령문
- 알려진 취약점 없음

*https://busybox.net

Chisel: 강화학습 기반 프로그램 축소 시스템†



Effective Program Debloating via Reinforcement Learning.
ACM Conference on Computer and Communications Security (***CCS***), 2018.
(컴퓨터 보안분야 최우수 국제학회)

† **http://chisel.cis.upenn.edu**

요구조건spec의 예

```
#!/bin/bash

function compile {
  clang -o tar.debloat tar-1.14.c
  return $?
}

# tests for the desired functionalities
function desired {
  # 1. archiving multiple files
  touch foo bar
  ./tar.debloat cf foo.tar foo bar
  rm foo bar
  ./tar.debloat xf foo.tar
  test -f foo -a -f bar || exit 1

  # 2. extracting from stdin
  touch foo
  ./tar.debloat cf foo.tar foo
  rm foo
  cat foo.tar | ./tar.debloat x
  test -f foo || exit 1

  # other tests
  …
  return 0
}
```

```
# tests for the undesired functionalities
function undesired {
  for test_script in `ls other_tests/*.sh`
  do
    { sh -x -e $test_script; } >& log
    grep 'Segmentation fault' log && exit 1
  done
  return 0
}

compile || exit 1
core || exit 1
non_core || exit 1
```

자동으로 축소된 tar-1.14 코드

사용되지 않는 변수 제거

```
int absolute_names;
int ignore_zeros_option;
struct tar_stat_info stat_info;

char *safer_name_suffix (char *file_name, int link_target) {
    int prefix_len;
    char *p;

    if (absolute_names) {
        p = file_name;
    } else {
       /* CVE-2016-6321 */
       /* Incorrect sanitization if “file_name” contains ".." */
       ...
    }
    ...
    return p;
}

void extract_archive() {
    char *file_name = safer_name_suffix(stat_info.file_name, 0);
    /* Overwrite “file_name” if exists */
    ...
}

void list_archive() { ... }
```

보안 취약점 CVE 내포된 코드 제거

사용되지 않는 기능 overwriting functionality
에 해당되는 부분 제거

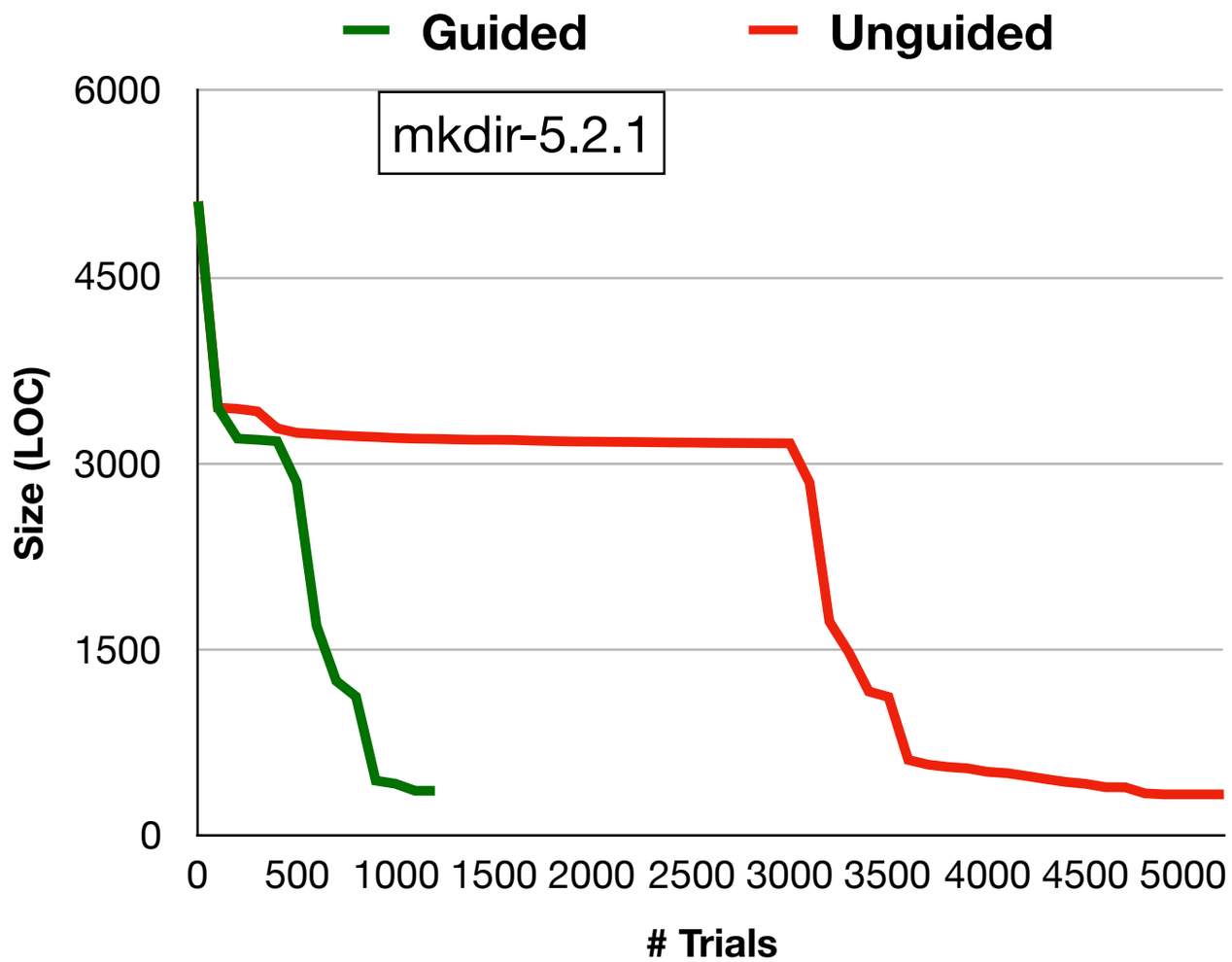
```
void read_and(void *(do_something)(void)) {
    enum read_header status;
    while (...) {
        status = read_header();
        switch (status) {
        case HEADER_SUCCESS: (*do_something)(); continue;
        case HEADER_ZERO_BLOCK:
          if (ignore_zeros_option) continue;
          else break;
        ...
        default: break;
        }
    }
    ...
}

/* Supports all options: -x, -t, -P, -i, ... */
int main(int argc, char **argv) {
    int optchar;
    while (optchar = getopt_long(argc, argv) != -1) {
        switch(optchar) {
        case 'x': read_and(&extract_archive); break;
        case 't': read_and(&list_archive); break;
        case 'P': absolute_names = 1; break;
        case 'i': ignore_zeros_option = 1; break;
        ...
        }
    }
    ...
}
```

불필요한 기능들에 해당되는 부분 제거

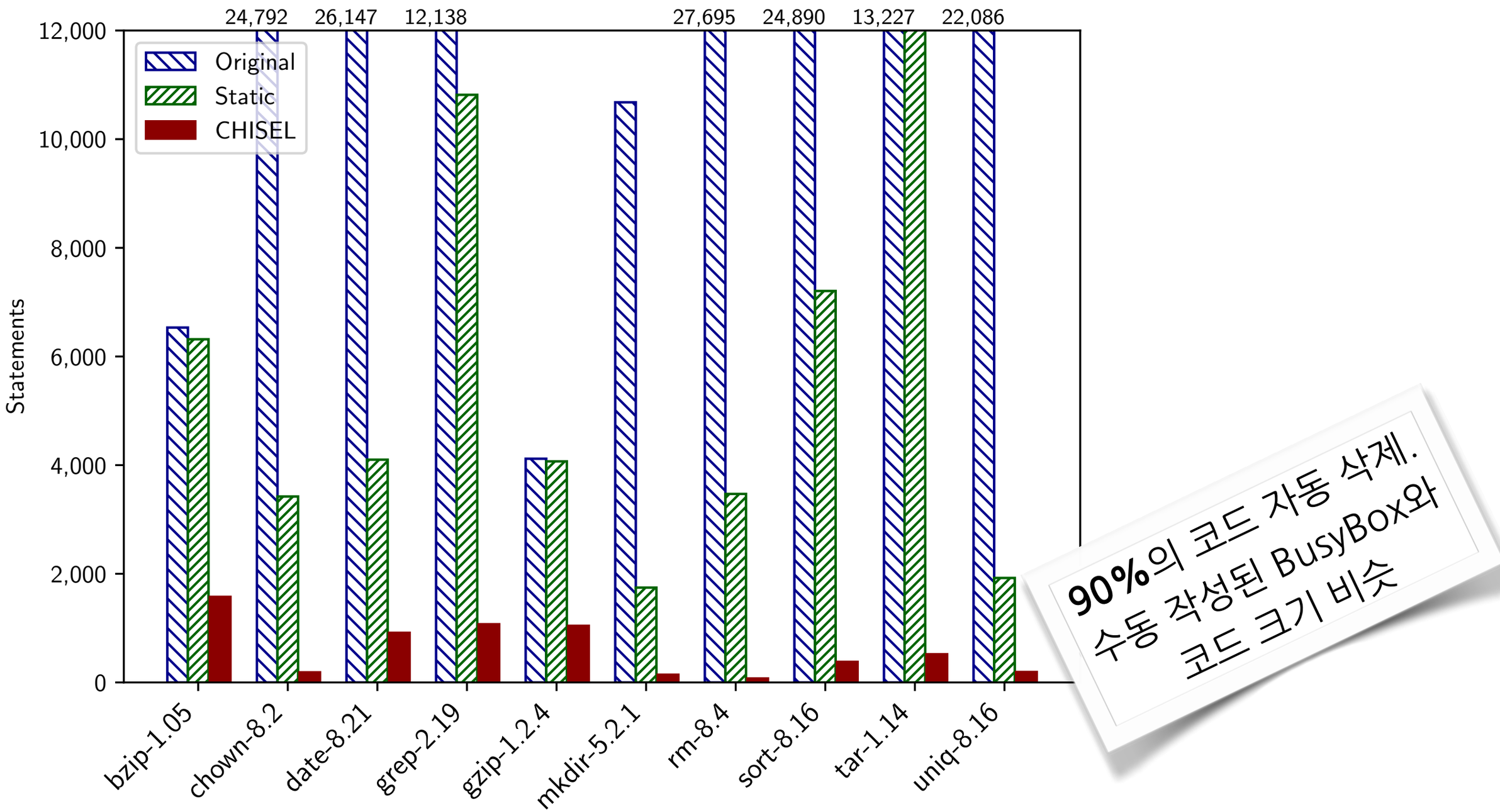
지원하지 않는 커맨드라인 옵션 제거

강화학습 기반 프로그램 축소 알고리즘

```
/* mkdir-5.2.1 */
int xstrtol(char *s, char **ptr, int strtol_base,
   strtol_t *val, char *valid_suffixes) {
 1: err = 0;
 2: assert(0 <= strtol_base && strtol_base <= 36);
 3: p = ptr ? ptr : &t_ptr;
 4: q = s;
 5: while(ISSPACE (*q)) ++q;
 6: if (*q == '-') return LONGINT_INVALID;
 7: errno = 0;
 8: tmp = strtol(s, p, strtol_base);
 9: if (*p == s) { … }
10: if (!valid_suffixes) { … }
11: if (**p != '\0') { … }
12: *val = tmp;
13: return err;
}
```



강화학습 기반 프로그램 축소 알고리즘



Datalog (확장된 SQL) 질의문 합성

- 목표: 하루에 수업을 두 개 이상 듣는 학생 명단을 뽑는 DB 질의문 생성

입력 테이블

Class

Class	Day
C1	Mon
C2	Mon
C3	Tue
C4	Tue
C5	Fri
C6	Fri

Enrolled

Student	Class Enrolled
S1	C1
S1	C2
S1	C3
…	…
S3	C2
S3	C5

출력 테이블

Busy

Student
S1
S6
S9
S10
S12
S13

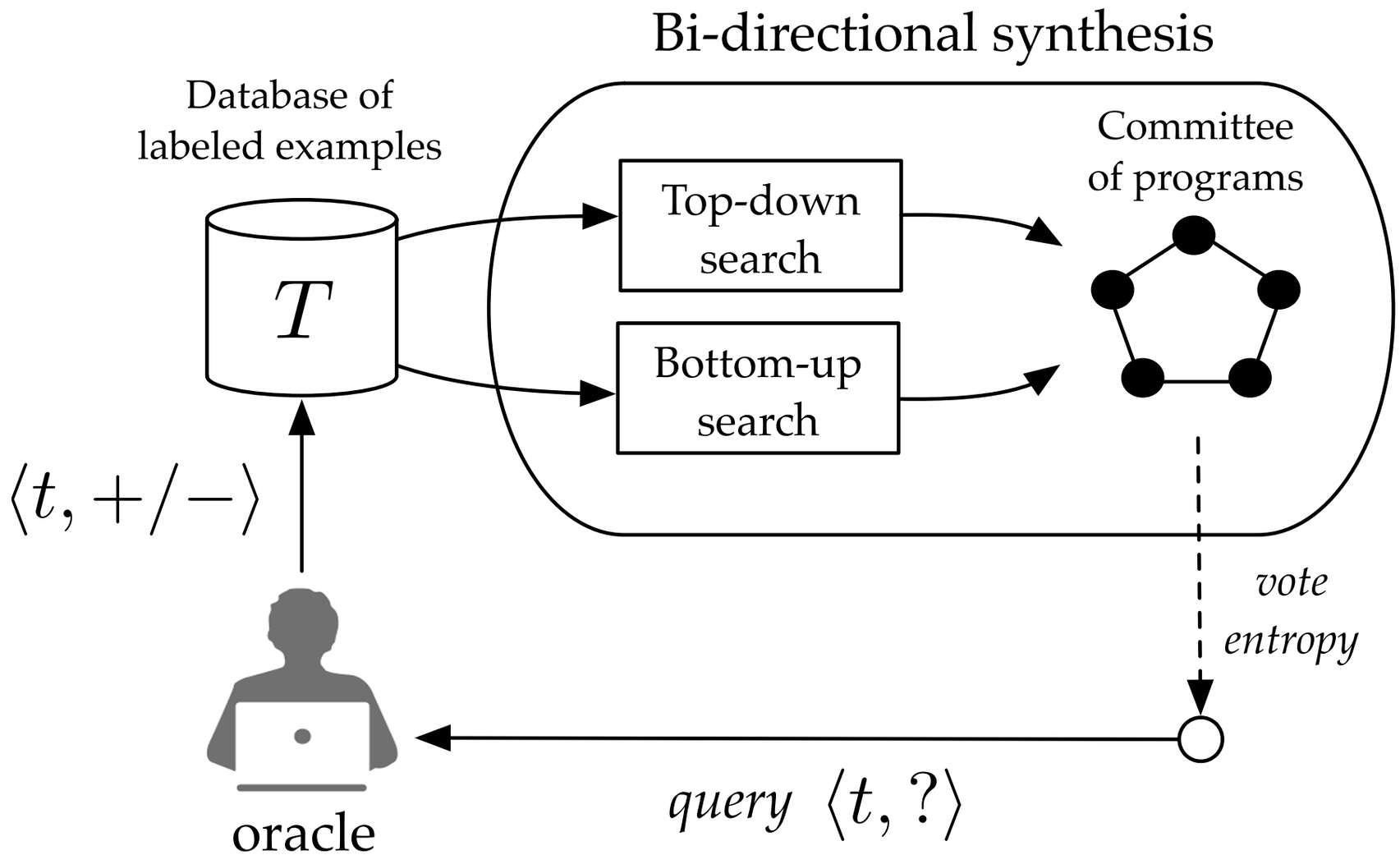
Datalog 질의문 자동 합성

```
EnrollClass(n, c, l) :- Enrolled(n, c), Class(c, l).
Busy(s) :- Student(s, n), EnrollClass(n, c1, l),
           EnrollClass(n, c2, l), c1 != c2.
```

= 다음의 SQL 질의문과 동일

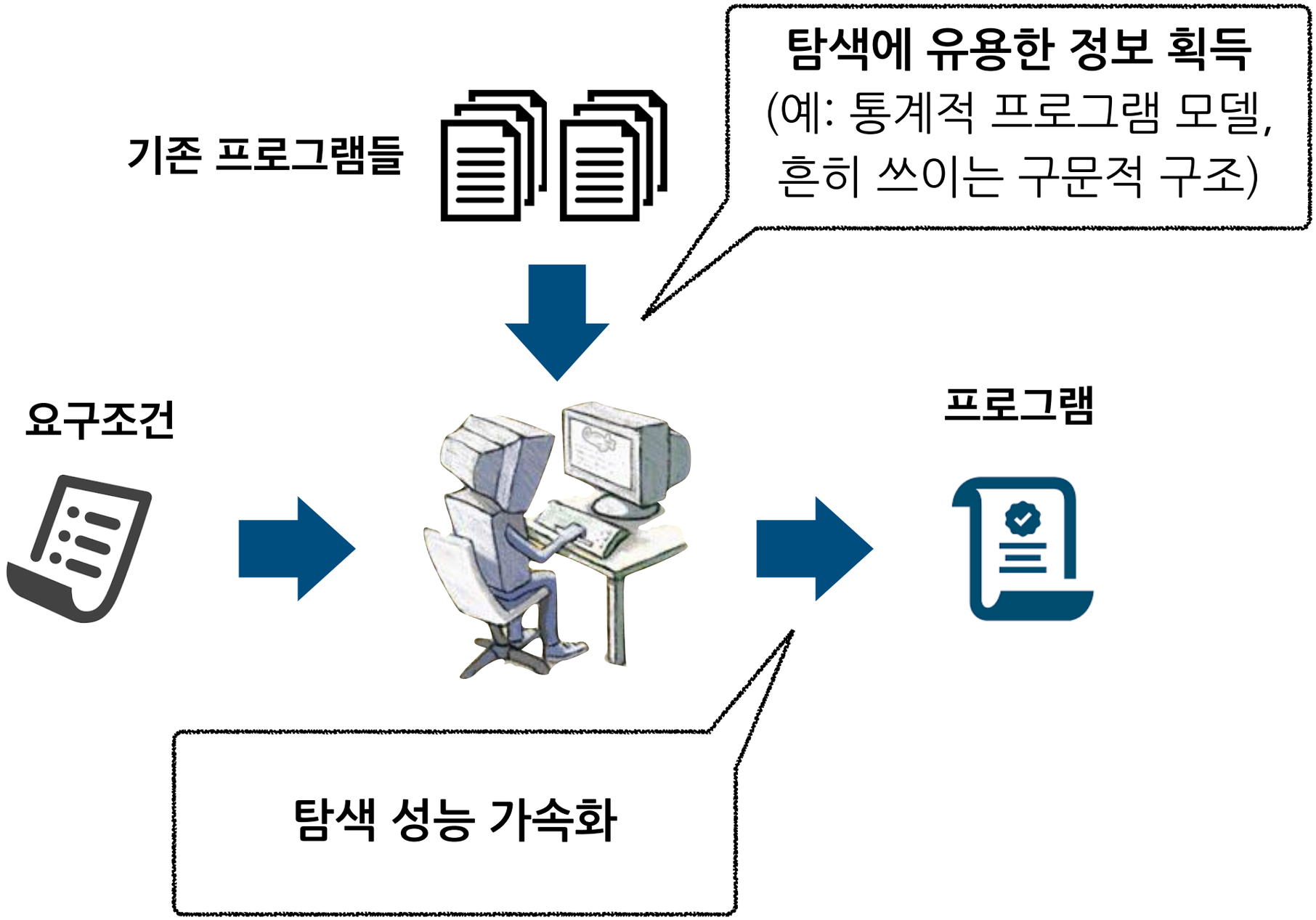
```
SELECT S.s FROM Student S
WHERE S.n IN (SELECT E1.n
                FROM Enrolled E1, Enrolled E2, Class C1, Class C2
                WHERE E1.n = E2.n AND E1.c <> E2.c
                AND E1.c = C1.c AND E2.c = C2.c AND C1.d = C2.d))
```

ALPS: Datalog 합성 프레임워크



Syntax-guided Datalog Synthesis.
ACM Conference on the Foundations of Software Engineering (***FSE***), 2018.
(소프트웨어 공학분야 최우수 국제학회)

요약


기존 프로그램들
탐색에 유용한 정보 획득
(예: 통계적 프로그램 모델,
흔히 쓰이는 구문적 구조)
요구조건
프로그램
탐색 성능 가속화