싹수 분석을 활용하여 비트 벡터 프로그램 양방향 합성 잘 하기

윤용호, 이우석, 이광근

2022 SIGPL 여름학교 번개발표 @ 전남대학교

윤용호, 서울대학교 프로그래밍연구실

요약

- 맥락: **프로그램 합성 문제**를 잘 풀고싶다
  - 자동 완성, 최적화 등의 분야에서 유용함
- 문제: 탐색 공간이 너무 너무 너무 너무 넓다
  - 불필요한 탐색을 줄이려는 다양한 시도가 진행중
  - SMT Solver가 널리 사용되지만 이 문제에 적용하기엔 너무 느림
- 해결책: 양방향 합성에 싹수 분석을 활용해서 탐색 공간을 확! 줄여보자
  - 정적 분석: 정확도 - 비용 조절 가능, 충분히 작은 프로그램이라면 아주 비싼 분석도 OK
- 평가: 잘 된다
  - SyGuS 벤치마크 중 해커의 기쁨(Hacker's Delight) 문제를 기존 합성기들보다 빨리 잘 푼다

맥락: 프로그램 합성 문제를 잘 풀고 싶다

- 프로그램 합성 = 조건에 맞는 프로그램을 자동으로 생성하기

- 유용함: 엑셀 자동 완성, LLVM souper(Super-optimizer), ...

- **Sy**ntax-**Gu**ided **S**ynthesis(SyGuS)가 표준처럼 널리 쓰임
  - SyGuS Language: 합성 문제를 표현하는 방법(문법+조건)
  - Benchmark: SyGuS Language로 표현된 각종 합성 문제

맥락: 프로그램 합성 문제를 잘 풀고 싶다

SyGuS Language

비트벡터 문제입니다

hd09 라는 이런 함수가 있는데

f: BitVec32 -> BitVec32
함수를 합성해주세요

이런 문법을 사용해주시고요

조건: hd09랑 f는 모든 입력에 대해
같은 값을 계산해야합니다

```
(set-logic BV)

(define-fun hd09
    ((x (BitVec 32)))
    (BitVec 32)
    (bvsub (bvxor x (bvashr x #x0000001F)) (bvashr x #x0000001F))
)

(synth-fun f
    ((x (BitVec 32)))
    (BitVec 32)
    ((Start (BitVec 32) ((bvnot Start)
                         (bvand Start Start)
                         (bvxor Start Start)
                         (bvor Start Start)
                         (bvneg Start)
                         (bvadd Start Start)
                         (bvmul Start Start)
                         (bvudiv Start Start)
                         (bvurem Start Start)
                         (bvlshr Start Start)
                         (bvashr Start Start)
                         (bvshl Start Start)
                         (bvsdiv Start Start)
                         (bvsrem Start Start)
                         (bvsub Start Start)
                         #x00000001
                         #x00000000
                         #x0000001F
                         #xFFFFFFFF
                         x)
    ))
)

(declare-var x (BitVec 32))
(constraint (= (hd09 x) (f x)))
(check-synth)
```

문제: 느려요

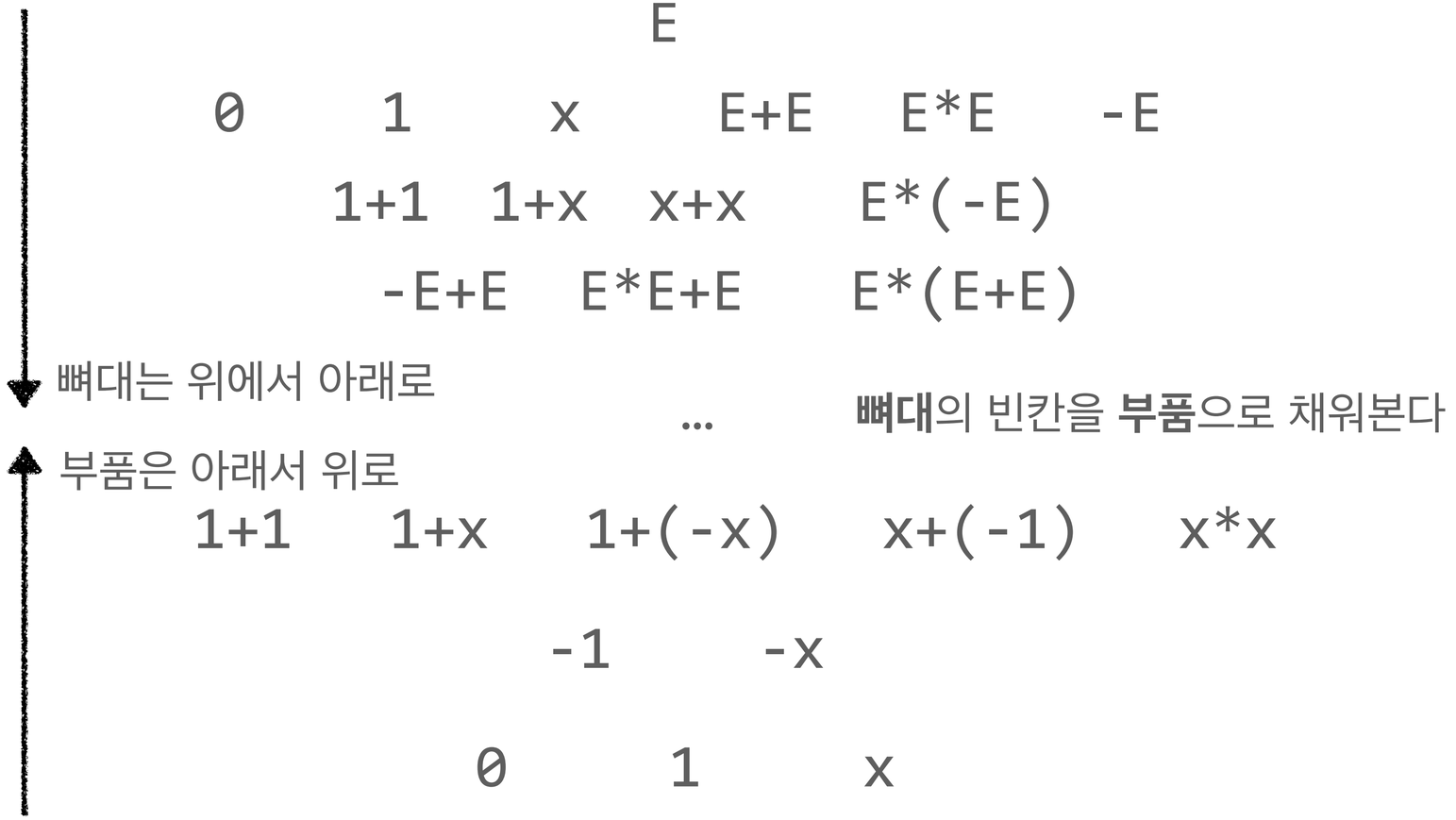
- 탐색 공간이 너무 너무 너무 너무 넓다
  - "주어진 언어로 작성할 수 있는 세상 모든 프로그램"

- 불필요한 탐색을 줄이고 정답에 빠르게 도달하기 위한 갖은 노력들
  - 양방향 합성* 등의 방법 존재 (이번 연구의 기반)

- SMT Solver가 보조 도구로 널리 사용되지만 여전히 느리다
  - 많이 빨라졌다곤 하지만 수만번 이상 호출하기엔 (1회당 0.x초)

또 맥락: 양방향 합성

Top-down + Bottom-up



$$
\begin{array}{rcl}
E & \rightarrow & 0 \\
 & | & 1 \\
 & | & x \\
 & | & E + E \\
 & | & E * E \\
 & | & -E
\end{array}
$$

Woosuk Lee, “Combining the Top-Down Propagation and Bottom-Up Enumeration for Inductive Program Synthesis”, POPL'21

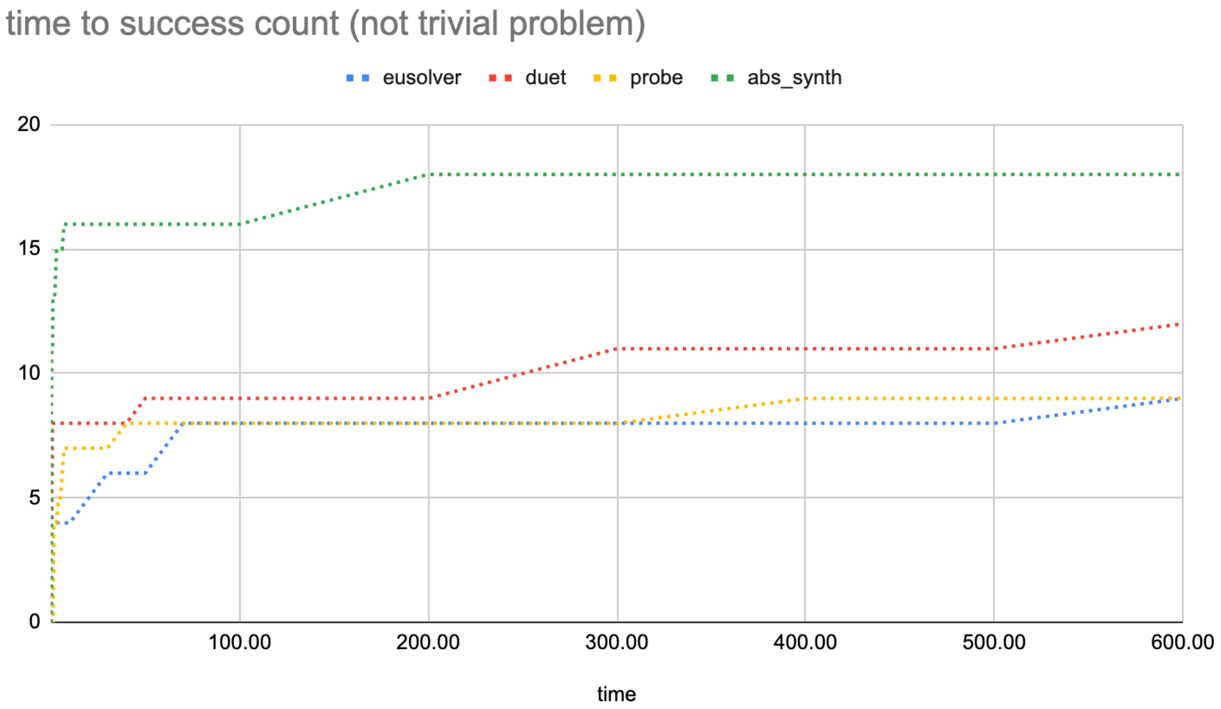
해결책: "싹수 분석"을 활용

- 합성이 아직 끝나지 않은 미완성 프로그램을 **정적 분석**
  - **싹수**가 없다: 이 프로그램을 마저 완성해봐야 절대 합성 조건을 만족할 수 없다
    - 싹수 없는 프로그램은 탐색 중단
  - 정적 분석 비용이 탐색 비용보다 충분히 저렴하면 성공

- 정적 분석의 장점은 적극 활용: 설계에 따라선 극한의 정확도를 뽑아낼 수 있다
- 정적 분석의 단점은 덜 드러남: 합성 대상 프로그램의 규모가 작아서 분석 비용이 저렴하다

평가: 잘 한다!

- 21*개의 해커의 기쁨 벤치마크 대상, 싹수 분석 기반 합성 도구를 기존 3가지 도구와 비교 eusolver[1], duet[2], probe[3]



*44개 문제 중 단순 나열로도 금방 답이 나오는 쉬운 문제 23개 제외, 원본의 32비트 문제를 64비트 문제로 확장함

[1] eusolver: Rajeev Alur et al., "Scaling Enumerative Program Synthesis via Divided and Conquer, TACAS'17
[2] duet: Woosuk Lee, POPL'21
[3] probe: Shraddha Barke et al., "Just-in-time learning for bottom-up enumerative synthesis", OOPSLA'20

결론

- 정적 분석과 양방향 합성의 만남으로 비트 벡터 합성 문제를 아주 잘 풀 수 있었다

- 확장 가능
  - 비트 벡터 도메인 외에도 써킷 도메인으로 확장 진행중

- 자세한 내용을 원하신다면 포스터 앞에서 만나요!